Learning Commonalities in RDF and SPARQL
Sara El Hassad, François Goasdoué, Hélène Jaudoin

To cite this version:
Sara El Hassad, François Goasdoué, Hélène Jaudoin. Learning Commonalities in RDF and SPARQL. [Research Report] Université Rennes 1. 2016. hal-01386237v4

HAL Id: hal-01386237
https://inria.hal.science/hal-01386237v4
Submitted on 23 May 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
Learning Commonalities in RDF and SPARQL
v4 of the Research Report made available online in early September 2016 (v1).

The main change in v2 was the addition of Section 5. In v3, some results and discussions were added to the core Sections 3 and 4; also details were added to the description of the related work in Section 7. In v4, experiments with DBpedia were added to Section 6.

Sara El Hassad  
IRISA, Univ. Rennes 1  
Lannion, France  
sara.el-hassad@irisa.fr

François Goasdoué  
IRISA, Univ. Rennes 1  
Lannion, France  
fg@irisa.fr

Hélène Jaudoin  
IRISA, Univ. Rennes 1  
Lannion, France  
helene.jaudoin@irisa.fr

ABSTRACT
Finding commonalities between descriptions of data or knowledge is a fundamental task in Machine Learning. The formal notion characterizing precisely such commonalities is known as least general generalization of descriptions and was introduced by G. Plotkin in the early 70’s, in First Order Logic.

Identifying least general generalizations has a large scope of database applications ranging from query optimization (e.g., to share commonalities between queries in view selection or multi-query optimization) to recommendation in social networks (e.g., to establish connections between users based on their commonalities between profiles or searches).

To the best of our knowledge, this is the first work that revisits the notion of least general generalizations in the entire Resource Description Framework (RDF) and popular conjunctive fragment of SPARQL, a.k.a. Basic Graph Pattern (BGP) queries. Our contributions include the definition and the computation of least general generalizations in these two settings, which amounts to finding the largest set of commonalities between incomplete databases and conjunctive queries, under deductive constraints. We also provide an experimental assessment of our technical contributions.

1. INTRODUCTION
Finding the commonalities between descriptions of data or knowledge is a foundational reasoning problem of Machine Learning, which was formalized in the early 70’s as computing a least general generalization \( \lgg \) of such descriptions \([39, 40]\). Since the early 90’s, this problem has also received consideration in the Knowledge Representation field, where least general generalization were rebaptized least common subsumers \([18]\), in Description Logics, e.g., \([18, 10, 11, 50]\) and in Conceptual Graphs \([17]\).

Interestingly, this problem has a large scope of applications when transposed in a database setting, i.e., when descriptions are expressed in a given data model or query language. In Query Optimization, \( \lgg \)s of subsets of a query workload \( \mathcal{W} \) may be used to identify candidate views or potential query sharing, with which \( \mathcal{W} \) can be efficiently evaluated, in the View Selection problem or in the Multi-Query Optimization problem, respectively. In Recommendation, in particular in a social context, \( \lgg \)s of sets of user descriptions (i.e., profiles) may help recommending users other users, or to form a community, when they share enough interests. Also, \( \lgg \)s of sets of queries issued by distinct users may help recommending users to each other, if what they ask for is enough related. In Exploration, \( \lgg \)s of datasets may be used to classify/categorize them w.r.t. their common information, to identify common social graph patterns between organizations (e.g., criminal ones), or to help identifying new potential links between datasets in the Linked Data Cloud.

Contributions. In this work, we revisit the problem of computing least general generalizations in the RDF data model and its SPARQL query language, the prominent Semantic Web standards of the W3C. This is a necessary first step towards using \( \lgg \)s within central database problems. More precisely, our contributions to this problem are:

1. We define and study the problem of computing an \( \lgg \) of RDF graphs in the entire RDF standard: we do not restrict RDF graphs in any way, i.e., neither their structure nor their semantics defined upon RDF entailment (inference). The recent work \([20, 19]\) brings a limited solution to the problem. It allows finding the commonalities between single entities extracted from RDF graphs (e.g., users in a social network), ignoring RDF entailment. In contrast, we further aim at considering the problem in all its generality, i.e., finding the commonalities between general RDF graphs, hence modeling multiple interrelated entities (e.g., social networks of users), accurately w.r.t. their standard semantics.

2. We define and study the problem of computing an \( \lgg \) of general SPARQL conjunctive queries, a.k.a. Basic Graph Pattern Queries (BGPQs), while the literature only considers unary tree-shaped conjunctive queries (UTCQ) \([31]\). Further, when available, we devise how to use background knowledge, formalized as ontological constraints modeling the application domain, in order to compute much more pregnant \( \lgg \)s.

3. We provide algorithms for our solutions to these two
problems. In particular, since RDF graphs may be large, we provide algorithms that allow computing *lggs* of small-to-huge general RDF graphs (i.e., that fit either in memory, in data management systems or in MapReduce clusters) w.r.t. any set of entailment rules from the RDF standard.

4. We provide experiments using synthetic LUBM data and real DBpedia data, which show to which extent *lggs* of BGPQs are much more present when background knowledge is available.

The paper is organized as follows. In Section 2, we introduce the RDF data model and its SPARQL query language. Then, we study the problem of computing an *lgg* of RDF graphs in Section 3, and of BGPQs in Section 4. In Section 5, we present algorithms for the problems studied in the two preceding Sections. We report on the experiments we conducted in Section 6. Finally, we present related works in Section 7 and we conclude and draw some perspectives in Section 8.

2. PRELIMINARIES

We introduce the RDF data model in Section 2.1. Then, in Section 2.2, we present the SPARQL conjunctive queries, a.k.a. Basic Graph Pattern queries (BGPQs).

2.1 Resource Description Framework (RDF)

2.1.1 RDF graphs

The RDF data model allows specifying RDF graphs. An RDF graph is a set of triples of the form (a, p, o). A triple states that its subject a has the property p, the value of which is the object o. Triples are built using three pairwise disjoint sets: a set U of uniform resources identifiers (URIs), a set L of literals (constants), and a set B of blank nodes allowing to support incomplete information. Blank nodes are local identifiers for missing values within an RDF graph (unknown URIs or literals); they are the RDF counterparts of labelled nulls or existential variables in incomplete relational databases [28, 9], as shown in [25]. Well-formed triples, as per the RDF specification [46], belong to \((U \cup B) \times U \times (U \cup L \cup B)\); we only consider such triples hereafter.

**Notations.** We use a, p, o in triples as placeholders. We note \(Val(\mathcal{G})\) the set of values occurring in an RDF graph \(\mathcal{G}\), i.e., the URIs, literals and blank nodes; we note \(Bl(\mathcal{G})\) the set of blank nodes occurring in \(\mathcal{G}\). A blank node is written b possibly with a subscript, and a literal is a string between quotes. For instance, the triples (b, hasTitle, “LGG in RDF”) and (b, hasContactAuthor, h1) state that something (b) entitled “LGG in RDF” has somebody (h1) as contact author.

A triple model an assertion, either for a class (unary relation) or for a property (binary relation). Table 1 (top) shows the use of triples to state such assertions, as well as the relational assertions to which they correspond. The RDF standard [46] provides built-in classes and properties, as URIs within the rdf and rdfs pre-defined namespaces, e.g., rdf:type which can be used to state that the above b is a conference paper with the triple (b, rdf:type, ConfPaper).

2.1.2 Adding ontological knowledge to RDF graphs

<table>
<thead>
<tr>
<th>RDF statement</th>
<th>Triple</th>
<th>Relational assertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class assertion</td>
<td>(a, rdf:type, o)</td>
<td>o(e)</td>
</tr>
<tr>
<td>Property assertion</td>
<td>(a, p, o) with (p \neq) rdf:type</td>
<td>p(a, o)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RDFS statement</th>
<th>Triple</th>
<th>Relational constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclass</td>
<td>(a, rdfs:subClassOf, o)</td>
<td>e \subseteq o</td>
</tr>
<tr>
<td>Subproperty</td>
<td>(a, rdfs:subPropertyOf, o)</td>
<td>e \subseteq o</td>
</tr>
<tr>
<td>Domain typing</td>
<td>(a, rdfs:domain, e)</td>
<td>\Pi_{\text{domain}(e)}(o) \subseteq o</td>
</tr>
<tr>
<td>Range typing</td>
<td>(a, rdfs:range, o)</td>
<td>\Pi_{\text{range}(e)}(o) \subseteq o</td>
</tr>
</tbody>
</table>

Table 1: RDF (top) & RDFS (bottom) statements.

Figure 1: Sample RDF graph \(\mathcal{G}\).

An essential feature of RDF is the possibility to enhance the descriptions in RDF graphs by declaring ontological constraints between the classes and properties they use. This is achieved with RDF Schema (RDFS) statements, which are triples using particular built-in properties. Table 1 (bottom) lists the allowed constraints, the triples to state them, as well as the deductive relational constraints to which they correspond; domain and range denote respectively the first and second attribute of every property. For example, the triple (ConfPaper, rdfs:subClassOf, Publication) states that conference papers are publications, the triple (hasContactAuthor, rdfs:subPropertyOf, hasAuthor) states that having a contact author is having an author, the triple (hasAuthor, rdfs:domain, Publication) states that only publications have authors while the triple (hasAuthor, rdfs:range, Researcher) states that only researchers are authors of something.

**Notations.** From now, for conciseness, we use the following shorthands for built-in properties: \(\tau\) for rdf:type, \(\equiv_{sc}\) for rdfs:subClassOf, \(\equiv_{sp}\) for rdfs:subPropertyOf, \(\leftarrow_{d}\) for rdfs:domain, and \(\rightarrow_{r}\) for rdfs:range.

Figure 1 displays the usual representation of the RDF graph \(\mathcal{G}\) made of the seven above-mentioned triples, which are called the explicit triples of \(\mathcal{G}\). A triple (a, p, o) corresponds to the p-labeled directed edge from the a node to the o node. Explicit triples are shown as solid edges, while the implicit ones, which are derived using ontological constraints (see below), are shown as dashed edges.

Importantly, the semantics of ontological constraints differ from the typical relational database setting, where they are interpreted under the closed world assumption (CWA), i.e., as integrity constraints. In RDF, they are interpreted under the open world assumption (OWA), i.e., as deductive constraints. Under OWA, all the triples that can be deduced from the constraints, called implicit triples, are assumed to be part of the RDF graph even though they are not explicitly present in it. For instance, consider the RDF graph \(\mathcal{G}\) in Figure 1. The ontological constraint (hasContactAuthor, rdfs:subPropertyOf, hasAuthor) is violated under CWA, as
the triple \((b, \text{hasContactAuthor}, b_1)\) is explicit in \(G\) while \((b, \text{hasAuthor}, b_1)\) is not. By contrast, under OWA, the same triple, \((b, \text{hasContactAuthor}, b_1)\), making it implicit within \(G\). Further, this implicit triple together with the explicit ontological constraint \((\text{hasAuthor}, b_1)\) produces the explicit triple \((b_1, \text{rdf:type}, \text{Researcher})\).

### 2.1.3 Deriving the implicit triples of an RDF graph

The RDF standard defines a set of entailment rules in order to derive automatically all the triples that are implicit to an RDF graph. Table 2 shows the strict subset of these rules that we will use to illustrate important notions as well as our contributions in the next sections; importantly, our contributions hold for the whole set of entailment rules of the RDF standard, and any subset of thereof. The rules in Table 2 concern the derivation of implicit triples using ontological constraints (i.e., RDF statements). They encode the propagation of assertions through constraints (\text{rdfs2}, \text{rdfs3}, \text{rdfs7}, \text{rdfs9}), the transitivity of the \(\leq_{sp}\) and \(\leq_{sc}\) constraints (\text{rdfs5}, \text{rdfs11}), the complementation of domains or ranges through \(\leq_{sc}\) (\text{ext1}, \text{ext2}), and the inheritance of domains and of ranges through \(\leq_{sp}\) (\text{ext3}, \text{ext4}).

The saturation (a.k.a. closure) of an RDF graph \(G\) w.r.t. a set \(R\) of RDF entailment rules, is the RDF graph \(G^\infty\) obtained by adding to \(G\) all the implicit triples that can be derived from \(G\) using \(R\). Roughly speaking, the saturation \(G^\infty\) materializes the semantics of \(G\). It corresponds to the fixpoint obtained by repeatedly applying the rules in \(R\) to \(G\) in a forward-chaining fashion, a.k.a. chasing \(G\) with \(R\) in the database parlance. In RDF, the saturation is always finite and unique (up to blank node renaming), and does not contain implicit triples \([46, 47]\).

The saturation of the RDF graph \(G\) shown in Figure 1 corresponds to the RDF graph \(G^\infty\) in which all the \(G\) implicit triples have been made explicit. It is worth noting how, starting from \(G\), applying RDF entailment rules mechanizes the construction of \(G^\infty\). For instance, recall the reasoning sketched in Section 2.1.2 for deriving the triple \((b_1, \text{rdf:type}, \text{Researcher})\). This is automated by the following sequence of applications of RDF entailment rules: \((\text{hasContactAuthor}, \text{rdfs:subPropertyOf}, \text{hasAuthor})\) and \((b, \text{hasContactAuthor}, b_1)\). This triggers the \text{rdfs7} rule which adds \((b, \text{hasAuthor}, b_1)\) to the RDF graph. In turn, this new triple together with \((\text{hasAuthor}, \text{rdfs:range}, \text{Researcher})\) triggers the \text{rdfs3} rule which adds \((b_1, \text{rdf:type}, \text{Researcher})\).

### 2.1.4 Comparing RDF graphs

The RDF standard defines a generalization/specialization relationship between two RDF graphs, called entailment between graphs. Roughly speaking, an RDF graph \(G\) is more specific than another RDF graph \(G'\), or equivalently \(G'\) is more general than \(G\), whenever there is an embedding of \(G'\) into the saturation of \(G\), i.e., the complete set of triples that \(G\) models.

More formally, given any subset \(R\) of RDF entailment rules, an RDF graph \(G\) entails an RDF graph \(G'\), denoted \(G \models R G'\), iff there exists an homomorphism \(\phi\) from \(B_1(G')\) to \(\text{Val}(G^\infty)\) such that \([G']_\phi \subseteq G^\infty\), where \([G']_\phi\) is the RDF graph obtained from \(G'\) by replacing every blank node \(b\) by its image \(\phi(b)\).

Figure 2 shows an RDF graph \(G'\) that is entailed by the RDF graph \(G\) in Figure 1 w.r.t. the RDF entailment rules displayed in Table 2. In particular, \(G \models R G'\) holds for the homomorphism \(\phi\) such that: \(\phi(b) = b\) and \(\phi(b_2) = \text{"LGG in RDF"}\). By contrast, when \(R\) is empty, this is not the case (i.e., \(G \not\models R G'\)), as the dashed edges in \(G\) are not materialized by saturation, hence the \(G'\) triple \((b, \tau, \text{Publication})\) cannot have an image in \(G\) through some homomorphism.

**Notations.** When relevant to the discussion, we designate by \(G \models^\phi R G'\) the fact that the entailment \(G \models R G'\) holds due to the graph homomorphism \(\phi\). Also, when RDF entailment rules are disregarded, i.e., \(R = \emptyset\), we note the entailment relation \(\models\) (without indicating the rule set under consideration). Observe that, in this particular case, the entailment between RDF graphs collapses to the database notion of containment between two incomplete instances, each stored into a Triple(s,p,o) relation.

Importantly, some remarkable properties follow directly from the definition of entailment between two RDF graphs:

1. \(G\) and \(G^\infty\) are equivalent, noted \(G \equiv R G^\infty\), since clearly \(G \models R G^\infty\) and \(G^\infty \models R G\) hold.
2. \(G \models R G'\) iff \(G^\infty \models R G'\) holds.

In particular, the second above property points out how entailment reduces to standard database containment once saturation is computed.

### 2.2 SPARQL conjunctive queries

#### 2.2.1 Basic graph pattern queries

The well-established conjunctive fragment of SPARQL queries, a.k.a. basic graph pattern queries (BGPs), is the counterpart of the relational select-project-join queries; it is the most widely used subset of SPARQL queries in real-world applications \([36]\).

A Basic Graph Pattern (BGP) is a set of triple patterns, or simply triples by a slight abuse of language. They generalize RDF triples by allowing the use of variables. Given a set \(V\) of variables, pairwise disjoint with \(U, L\) and \(B\), triple patterns belong to: \((V \cup U \cup B) \times (V \cup U) \times (V \cup U \cup L \cup B)\).

**Notations.** We adopt the usual conjunctive query notation \(q(x) \leftarrow t_1, \ldots, t_o\), where \((t_1, \ldots, t_o)\) is a BGP; the query head variables \(x\) are called answer variables, and form a subset of the variables occurring in \(t_1, \ldots, t_o\); for boolean
queries, $\bar{x}$ is empty. The head of $q$, noted $\text{head}(q)$, is $q(\bar{x})$, and the body of $q$, denoted $\text{body}(q)$, is the BGP $\{t_1, \ldots, t_n\}$. We use $x$ and $y$ in queries, possibly with subscripts, for answer and non-answer variables respectively. Finally, we denote by $\text{VarB}(q)$ the set of variables and blank nodes occurring in the query $q$; we note $\text{Val}(q)$ the set of all its values, i.e., URIs, blank nodes, literals and variables.

2.2.2 Entailing and answering queries

Two important notions characterize how an RDF graph contributes to a query.

**Query entailment.** The weaker notion indicates whether or not an RDF graph holds some answer(s) to a query. It generalizes entailment between RDF graphs, to account for the presence of variables in the query body, for establishing whether a graph entails a query, i.e., whether the query embeds in that graph.

Formally, given a BGPQ $q$, an RDF graph $G$ and a set $\mathcal{R}$ of RDF entailment rules, $G$ entails $q$, denoted $G \models_\mathcal{R} q$, iff $G \models_\mathcal{R} \text{body}(q)$ holds, i.e., there exists a homomorphism $\phi$ from $\text{VarB}(q)$ to $\text{Val}(G^{\infty})$ such that $\text{body}(q)|_\phi \subseteq G^{\infty}$.

The RDF graph $G$ in Figure 1 entails the query $q(x_1, x_2) \leftarrow (x_1, \tau, x_2)$ asking for all the resources and their classes, for instance because of the homomorphism $\phi$ such that $\phi(x_1) = b$ and $\phi(x_2) = \text{ConfPaper}$. Observe that this entailment holds for any subset of RDF entailment rules, since the above $\phi$ already holds for $\mathcal{R} = \emptyset$, i.e., considering only the explicit triples in Figure 1.

**Notations.** Similarly to RDF entailment, we denote by $G \models_\mathcal{R} q$ that the entailment $G \models_\mathcal{R} q$ holds due to the homomorphism $\phi$.

**Query answering.** The stronger notion characterizing how an RDF graph contributes to a query, which builds on the preceding one, identifies all the query answers that the graph holds. Formally, assuming the head of a BGPQ $q$ is $q(\bar{x})$, the answer set of $q$ against $G$ is:

$$\text{q}(G) = \{(\bar{x})_\phi \mid G \models_\mathcal{R} \text{body}(q)\}$$

where we denote by $(\bar{x})_\phi$ the tuple of $G^{\infty}$ values obtained by replacing every answer variable $x_i \in \bar{x}$ by its image $\phi(x_i)$. Observe that when $\mathcal{R}$ is empty, BGPQ query answering coincide with standard relational conjunctive query evaluation.

The answer set to the above query $q(x_1, x_2) \leftarrow (x_1, \tau, x_2)$ against the RDF graph $G$ in Figure 1 is:

- $\{(b, \text{ConfPaper}), (b, \text{Publication}), (b_1, \text{Researcher})\}$ for $\mathcal{R}$ the set of entailment rules shown in Table 2, i.e., considering the explicit and implicit triples in Figure 1,
- $\{(b, \text{ConfPaper})\}$ for $\mathcal{R} = \emptyset$, i.e., considering only the explicit triples in Figure 1.

Importantly, from the definition of entailment between two RDF graphs [46, 47], the following holds:

**Property 1.** Given two RDF graphs $G, G'$ and a set $\mathcal{R}$ of RDF entailment rules, (i) $G$ and $G^{\infty}$ are equivalent ($G \models_\mathcal{R} G^{\infty}$ and $G^{\infty} \models_\mathcal{R} G$ hold), noted $G \equiv_\mathcal{R} G^{\infty}$, and (ii) $G \models_\mathcal{R} G'$ holds iff $G^{\infty} \models_\mathcal{R} G'$ holds.

From a practical viewpoint, Property 1 points out that checking $G \models_\mathcal{R} G'$ can be done in two steps: a reasoning step that computes the saturation $G^{\infty}$ of $G$, followed by a standard graph homomorphism step that checks if $G^{\infty} \models_\mathcal{R} G'$ holds.

2.2.3 Comparing queries

Similarly to RDF graphs, queries can be compared through the generalization/specialization relationship of entailment between queries. This relationship is the counterpart of that query containment in the relational database setting.

Let $q, q'$ be two BGPQs with same arity, whose heads are $q(\bar{x})$ and $q'(\bar{x}')$, and $\mathcal{R}$ the set of RDF entailment rules under consideration. $q$ entails $q'$, denoted $q \models_\mathcal{R} q'$, iff $\text{body}(q) \models_\mathcal{R} \text{body}(q')$ with $(\bar{x}')_\phi = \bar{x}$ holds. Here, $\text{body}(q) \models_\mathcal{R} \text{body}(q')$ is the adaptation of the above-mentioned entailment relationships between RDF graphs to the fact that the query bodies may feature variables, i.e., $\phi$ is a homomorphism from $\text{VarB}(q')$ to $\text{Val}(q)$ such that $\text{body}(q')|_\phi \subseteq \text{body}(q^{\infty})$; the saturation of a BGP, here $\text{body}(q)^{\infty}$, is the generalization of RDF graph saturation that treats variables as blank nodes, since they both equivalently model unknown information within BGPQs [46, 47].

For instance, the query $q_1(x)_:- (x, \tau, \text{ConfPaper}), (x, \text{hasContactAuthor}, y)$ entails the query $q_2(x)_:- (x, \tau, y)$ with $\phi(x) = x, \phi(y) = \text{ConfPaper}$ and any entailment rule set. However, it is worth noting that, counter-intuitively, $q_1$ does not entail the query $q_3(x)_:- (x, \tau, \text{Publication}), (x, \text{hasAuthor}, y)$. The reason is that query entailment does not consider extra ontological constraints, which would be needed by RDF entailment rules to find such elaborate entailment (here, that conference papers are publications, and that having a contact author is having an author). Extending standard entailment between BGPQs to take into account extra ontological constraints modeling background knowledge, is one contribution in this work (Section 4).

Remark that entailment between queries, query entailment and query answering (obviously) relate as follows:

**Property 2.** Given an RDF graph $G$, a set $\mathcal{R}$ of entailment rules and two BGPQs $q, q'$ such that $q \models_\mathcal{R} q'$, (i) if $G \models_\mathcal{R} q$ holds then $G \models_\mathcal{R} q'$ holds, and (ii) $q(G) \subseteq q'(G)$ holds.

Finally, remark that query entailment, query answering and entailment between queries treat the blank nodes in a query exactly as non-answer variables [48]. Therefore, from now on, we assume without loss of generality that queries do not use blank nodes.

3. FINDING COMMONALITIES BETWEEN RDF GRAPHS

In Section 3.1, we first define the largest set of commonalities between RDF graphs as a particular RDF graph representing their least general generalization (1gg for short). Then, we devise a technique for computing such an 1gg in Section 3.2.

3.1 Defining the 1gg of RDF graphs

A least general generalization of $n$ descriptions $d_1, \ldots, d_n$ is a most specific description $d$ generalizing every $d_i, 1 \leq i \leq n$, for some generalization/specialization relation between descriptions [39, 40]. In our RDF setting, we use RDF graphs as descriptions and entailment between RDF graphs as relation for generalization/specialization:

**Definition 1 (1gg of RDF graphs).** Let $G_1, \ldots, G_n$ be RDF graphs and $\mathcal{R}$ a set of RDF entailment rules.
A generalization of \(G_1, \ldots, G_n\) is an RDF graph \(G_g\) such that \(G_i \models R G_g\) holds for \(1 \leq i \leq n\).

A least general generalization (lgg) of \(G_1, \ldots, G_n\) is a generalization \(G_{lgg}\) of \(G_1, \ldots, G_n\) such that for any other generalization \(G_h\) of \(G_1, \ldots, G_n\), \(G_{lgg} \models R G_h\) holds.

Importantly, in the RDF setting, the following holds:

**Theorem 1.** An lgg of RDF graphs always exists, and is unique up to entailment.

**Proof.** An lgg of RDF graphs always exists, since we can always construct a (possibly empty) RDF graph that is the lgg of RDF graphs, in particular the cover graph of RDF graphs devised in Section 3.2. Also, an lgg of RDF graphs is unique up to entailment (since \(G_{lgg} \models R G_h\) holds for any \(G_h\) in Definition 1). Indeed, if it were that RDF graphs have *multiple lgg incomparable* w.r.t. entailment, say \(G_{lgg_1}, \ldots, G_{lgg_m}\), their merge\(^1\) \(G_{lgg_1} \cup \cdots \cup G_{lgg_m}\) would be a single strictly more specific lgg, a contradiction. \(\square\)

Figure 3 displays two RDF graphs \(G_1\) and \(G_2\), as well as their minimal lgg (with lowest number of triples) when we consider the RDF entailment rules shown in Table 2: \(G_{lgg}\). \(G_1\) describes a conference paper \(i_1\) with title “Disaggregations in Databases” and author Serge Abiteboul, who is a researcher; also conference papers are publications. \(G_2\) describes a journal paper \(i_2\) with title “Computing with First-Order Logic”, author contact Serge Abiteboul and author Victor Vianu, who are researchers; moreover, journal papers are publications and having a contact author is having an author. \(G_{lgg}\) states that their common information comprises the existence of a resource \((b_{i_1}, i_2)\) having some type \((bC(\text{conf}))\) and another \((bC(\text{journal}))\), which is a particular case of publication, with some title \((bd(id)(\text{conf})))\) and author Serge Abiteboul, who is a researcher.

Though unique up to entailment (i.e., semantically unique), an lgg may have many syntactical forms due to redundant triples. Such triples can be either explicit ones that could have been left implicit if the set of RDF entailment rules

\(^1\)The merge of RDF graphs, performed with the RDF specific merge operator \(\cup\), is an RDF graph comprising the union of the input RDF graph after renaming their blank nodes with fresh ones, so that these RDF graphs do not share (thus join on) such values. Indeed, blank nodes are used to characterize the incompleteness of an RDF graph, hence are local to it (Section 2).
Clearly, $G_{\text{gg}}^2$ is a possible value for $G$ above, because $G_{1 \leq n} \models \emptyset$ $G_{\text{gg}}^2$ implies $G_{1 \leq n} \models \emptyset$ $G_{\text{gg}}^2$ hence $G_{\text{gg}}^2$ must hold. <

Now, let us prove Proposition 1 using the above lemma.

(⇒) Assume that $G_{\text{gg}}$ is an 1gg of $G_1, \ldots, G_n$ and let us show that it is also an 1gg of some 1gg of $G_1, \ldots, G_{n-1}$ and $G_n$. By Definition 1, because $G_{\text{gg}}$ is an 1gg of $G_1, \ldots, G_n$, we have: (i) $G_{1 \leq n} \models \emptyset$ $G_{\text{gg}}$ holds and (ii) for any RDF graph $G$ such that $G_{1 \leq n} \models \emptyset$ $G$ holds, $G_{\text{gg}} \models \emptyset$ $G$ holds.

Let $G_{\text{gg}}$ be some 1gg of $G_1, \ldots, G_{n-1}$. From (i) and the above lemma, (*) $G_{\text{gg}} \models \emptyset$ $G_{1 \leq n}$ and $G_{\text{gg}} \models \emptyset$ $G_n$ holds.

Moreover, for any RDF graph $G$, if it were that $G_{\text{gg}} \models \emptyset$ $G$ and $G_n \models \emptyset$ $G$ and $G_{\text{gg}} \models \emptyset$ $G$, then $G_{1 \leq n} \models \emptyset$ $G$ would hold and contradict the fact that $G_{\text{gg}}$ is an 1gg of $G_1, \ldots, G_n$. Therefore, (***) for any RDF graph $G$, if $G_{\text{gg}} \models \emptyset$ $G$ and $G_n \models \emptyset$ $G$ holds, then $G_{\text{gg}} \models \emptyset$ $G$ holds.

From Definition 1, (*) and (***) we get that $G_{\text{gg}}$ is an 1gg of an 1gg of $G_1, \ldots, G_{n-1}$ and $G_n$.

(⇐) Assume that $G_{\text{gg}}$ is an 1gg of an 1gg $G'$ of $G_1, \ldots, G_{n-1}$ and $G_n$, and let us show that it is also an 1gg of $G_1, \ldots, G_n$.

By Definition 1, (i) $G' \models \emptyset$ $G_{\text{gg}}$ and $G_n \models \emptyset$ $G_{\text{gg}}$ hold, hence $G_{1 \leq n-1} \models \emptyset$ $G_{\text{gg}}$ and $G_n \models \emptyset$ $G_{\text{gg}}$ hold, i.e., (***) $G_{1 \leq n-1} \models \emptyset$ $G_{\text{gg}}$ holds.

Moreover, (ii) for any RDF graph $G$ such that $G' \models \emptyset$ $G$ and $G_n \models \emptyset$ $G$ hold, $G_{\text{gg}} \models \emptyset$ $G$ holds. By Definition 1, $G_{1 \leq n-1} \models \emptyset$ $G'$ holds, therefore we get: (***) for any RDF graph $G$ such that $G_{1 \leq n-1} \models \emptyset$ $G'$ holds, $G_{\text{gg}} \models \emptyset$ $G$ holds.

From Definition 1, (***) and (***) we get that $G_{\text{gg}}$ is an 1gg of $G_1, \ldots, G_n$. □

Based on the above result, without loss of generality, we focus in the next section on the following problem:

**Problem 1.** Given two RDF graphs $G_1, G_2$ and a set $R$ of RDF entailment rules, we want to compute some 1gg of $G_1$ and $G_2$.

### 3.2 Computing an 1gg of RDF graphs

We first devise the cover graph of two RDF graphs $G_1$ and $G_2$ (to be defined shortly, Definition 2 below), which is central to our technique for computing an 1gg of $G_1$ and $G_2$. We indeed show (Theorem 2) that this particular RDF graph corresponds to an 1gg of $G_1$ and $G_2$ when considering their explicit triples only, i.e., ignoring RDF entailment rules. Then, we show the main result of this section (Theorem 3): an 1gg of $G_1$ and $G_2$, for any set $R$ of RDF entailment rules, is the cover graph of their saturations w.r.t. $R$. We also provide the worst-case size of cover graph-based 1ggs, as well as the worst-case time to compute them.

**Definition 2 (Cover Graph).** The cover graph $G$ of two RDF graph $G_1$ and $G_2$ is the RDF graph such that for every property $p$ in both $G_1$ and $G_2$:

$$\{t_1, p, t_2\} \in G_1 \text{ and } (t_3, p, t_4) \in G_2 \text{ iff } (t_5, p, t_6) \in G$$

with $t_5 = t_1$ if $t_1 = t_3$ and $t_1 \in U \cup U$, else $t_5$ is the blank node $b_{t_1, t_3}$, and, similarly $t_6 = t_2$ if $t_2 = t_4$ and $t_2 \in U \cup U$, else $t_6$ is the blank node $b_{t_2, t_4}$.

The cover graph is more general than $G_1$ and $G_2$ as each of its triple $(t_1, p, t_3)$ is a least general uniﬁer of a triple $(t_1, p, t_3)$ from $G_1$ and a triple $(t_3, p, t_4)$ from $G_2$. The notion of least general uniﬁer [39, 40, 43] is dual to the well-known notion of most general uniﬁer [42, 43]. Observe that $G$’s triples result from anti-unifications of $G_1$ and $G_2$ triples with same property URI. Indeed, anti-unifying triples of the form $(s_1, p, o_1)$ and $(s_2, p', o_2)$, with $p \neq p'$, would lead to a non-well-formed triple of the form $(s, b_{p, p'}, o)$ (recall that property values must be URIs in RDF graphs), where $b_{p, p'}$ is the blank node required to generalize the distinct values $p$ and $p'$.

Further, the cover graph is an 1gg for the explicit triples in $G_1$ and those in $G_2$ since, intuitively, we capture their common structures by consistently naming, across all the anti-unifications begetting $G$, the blank nodes used to generalize pairs of distinct subject values or of object values: each time the distinct values $t$ from $G_1$ and $t'$ from $G_2$ are generalized by a blank node while anti-unifying two triples, it is always by the same blank node $b_{t, t'}$ in $G$. This way, we establish joins between RDF graphs, which reflect the common join structure on $t$ within $G_1$ and on $t'$ within $G_2$. For instance in Figure 3, the explicit triples $(i_1, \tau, \text{ConPaper})$, $(\text{ConPaper}, \tau, \text{Publication})$, $(i_1, \text{title}, \text{"DiD"})$ in $G_1$, and $(i_2, \tau, \text{JourPaper})$, $(\text{JourPaper}, \tau, \text{Publication})$, $(i_2, \text{title}, \text{"CwFOL"})$ in $G_2$, lead to the triples $(b_{i_1, \tau}, \tau, \text{BCPJP})$, $(b_{\text{BCPJP}}, \tau, \text{Publication})$, $(b_{i_2, \text{title}}, b_{\text{BCPJP}}, \text{DC})$ in the cover graph of $G_1$ and $G_2$ shown in Figure 4 (left). The first above-mentioned $G$ triple results from anti-unifying $i_1$ and $i_2$ into $b_{i_1, i_2}$, and, ConPaper and JourPaper into $b_{\text{BCPJP}}$. The second results from anti-unifying against ConPaper and JourPaper into $b_{\text{BCPJP}}$, and Publication and Publication into Publication (as a constant is its own least general generalization). Finally, the third results from anti-unifying against $i_1$ and $i_2$ into $b_{i_1, i_2}$, and, "DiD" and "CwFOL" into $b_{\text{DC}}$. By reusing consistently the same blank node name $b_{i_1, i_2}$ for each anti-unification of the constants $i_1$ and $i_2$ (resp. $b_{\text{BCPJP}}$ for ConPaper and JourPaper), the cover graph triples join on $b_{i_1, i_2}$ (resp. $b_{\text{BCPJP}}$) in order to reflect that, in $G_1$ and in $G_2$, there exists a particular case of publication $(i_1$ in $G_1$ and $i_2$ in $G_2$) with some title ("DiD" in $G_1$ and "CwFOL" in $G_2$).

The next theorem formalizes the above discussion by stating that the cover graph of two RDF graphs is an 1gg of them, just in case of an empty set of RDF entailment rules.

**Theorem 2.** The cover graph $G$ of the RDF graphs $G_1$ and $G_2$ is an 1gg of them for the empty set $R$ of RDF entailment rules (i.e., $R = \emptyset$).

**Proof.** The proof can be directly derived from that of the more general Theorem 3 (see below), noting that in the particular case of Theorem 2 $R = \emptyset$ holds, hence $G_1^\infty = G_1$ and $G_2^\infty = G_2$ holds. □

We provide below worst-case bounds for the time to compute a cover graph and for its size; these bounds are met when all the triples of the two input graphs use the same property URI (i.e., every pair of $G_1$ and $G_2$ triples begets a $G$ triple).

**Proposition 2.** The cover graph of two RDF graphs $G_1$ and $G_2$ can be computed in $O(|G_1| \times |G_2|)$; its size is bounded by $|G_1| \times |G_2|$.

The main theorem below generalizes Theorem 2 in order to take into account any set of entailment rules from the RDF standard. It states that it is sufficient to compute the cover of the saturations of the input RDF graphs, instead of the input RDF graphs themselves.
Theorem 3. Let $G_1$ and $G_2$ be two RDF graphs, and $R$ a set of RDF entailment rules. The cover graph $G$ of $G_1$ and $G_2$ is an 1gg of $G_1$ and $G_2$.

Proof. We start by showing that $G$ is a generalization of $G_1$ and $G_2$, i.e., $G_1 \models_R G$ and that $G_2 \models_R G$. Consider the RDF graph $G'$ obtained from $G$ by replacing every blank node $b_{i1,v_2}$ by the value $v_1$. Clearly, $G' \models_R G$ and, by construction of $G$, $G' = G_1$, i.e., $G_1 \models_R G$. Showing $G_2 \models_R G$ is done in a similar way. Hence, $G$ is a generalization of $G_1$ and $G_2$.

Let us show now that $G$ is an 1gg of $G_1$ and $G_2$. Let $G_{1gg}$ be any 1gg of $G_1$ and $G_2$. To prove our claim, we need to show that $G \models_R G_{1gg}$ holds. Since $G_{1gg}$ is an 1gg of $G_1$ and $G_2$, there exist two homomorphisms $\phi_1$ and $\phi_2$ from the blank nodes in $G_{1gg}$ to the values in $G_1^n$ and in $G_2^n$ respectively, such that $[G_{1gg}]_{\phi_1} \subseteq G_1^n$ and $[G_{1gg}]_{\phi_2} \subseteq G_2^n$. Consider the graph $G_{1gg}'$ obtained from $G_{1gg}$ by replacing every blank node $b$ by $b_{1,v_2}$ where $v_1 = \phi_1(b)$ and $v_2 = \phi_2(b)$. Clearly, $G_{1gg} \equiv_R G_{1gg}'$ holds. Let us show that $G \models_R G_{1gg}'$ holds, i.e., there exists a homomorphism $\phi$ from the blank nodes in $G_{1gg}'$ to the terms in $G^n$ such that $[G_{1gg}']_{\phi} \subseteq G^n$. By construction of $G$, it is easy to see that the above holds when $\phi$ maps $b_{i1,v_2}$ either to $v_1 = v_2$ and $v_1 \in U \cup L$, or to itself otherwise. It therefore follows that $G \models_R G_{1gg}'$, hence $G \models_R G_{1gg}$.

As an immediate consequence of the above results, we get the following worst-case bounds for the time to compute a cover graph-based 1gg of two RDF graphs $G_1$ and $G_2$, and for its size. Here, we assume given the saturation $G_1^n$ and $G_2^n$, as the times to compute them and their sizes depend on the RDF entailment rules at hand.

Corollary 1. An 1gg of two RDF graphs $G_1$ and $G_2$ can be computed in $O(|G_1^n| \times |G_2^n|)$ and its size is bounded by $|G_1^n| \times |G_2^n|$.

Remark that computing naively the cover graph-based 1gg of $n$ RDF graphs of size $M$ based on Proposition 1 may lead to an 1gg of size $M^n$, in the unlikely worst-case where all the triples of all the RDF graphs use the same property URL. However, removing the redundant triples from the intermediate and final cover graph-based 1ggs limits their size to at most $M$.

Figure 4 (right) displays an 1gg of the RDF graphs $G_1$ and $G_2$ in Figure 3 w.r.t. the entailment rules shown in Table 2. In contrast to Figure 4 (left), which shows an 1gg of the same RDF graphs when RDF entailment rules are ignored, we further learn that Serge Abiteboul is an author of some particular publication ($i_1$ in $G_1$ and $i_2$ in $G_2$). Moreover, removing the redundant triples (the grey ones) yields precisely the $1gg G_{1gg}$ of $G_1$ and $G_2$ shown in Figure 3.

4. Finding Commonalities Between Queries

We consider now the problem of finding the largest set of commonalities between queries. Adapting the results for RDF graphs from the preceding section to BGPQs is rather direct (recall that a query body is a BGP, i.e., an RDF graph in which variables can be used in subject, property and object positions of triples), but of limited interest as the next example shows. Consider the two BGPQs $q_1$ and $q_2$ in Figure 6. The first asks for the conference papers having some contact author, while the second asks for the journal papers having some author. Clearly, a least general generalization of $q_1$ and $q_2$ is the very general BGPQ $q_{1gg}$ shown in Figure 6 that asks for the resources having some type. Observe that $q_{1gg}$ is an 1gg for any subset of the RDF entailment rules shown in Table 2, as there is no ontological constraint in the queries. However, by considering the extra ontological constraints displayed in Figure 7 that hold in the scientific publication domain, i.e., the context in which the queries are asked, a more pregnant 1gg would be a BGPQ asking for publications having some researcher as author, since (i) having a contact authors is having an author, (ii) only publications have authors, (iii) only researchers are authors, and (iv) conference (resp. journal) papers are publications.

We therefore devise in Section 4.1 a notion of 1gg of BGPQs in the presence of ontological constraints, which as exemplified above helps finding more interesting commonalities by taking into account the background knowledge of an application domain. Then, we provide a technique for computing such an 1gg of BGPQs in Section 4.2.

4.1 Defining the 1gg of BGPQs

Recall that the notion of 1gg of $n$ descriptions $d_1, \ldots, d_n$ is a most specific description $d$ generalizing $d_1, \ldots, d_n$ for some generalization/specialization relation. Here, we adapt this notion by using BGPQs as descriptions and revisiting en-
 entailment between BGPQs (Section 2) in order to endow this RDF relation between queries with background knowledge.

We first devise in Section 4.1.1 entailment between BGPQs w.r.t. ontological constraints as a generalization of the standard entailment between BGPQs. Then, we define the 1gg of BGPQs w.r.t. ontological constraints in Section 4.1.2.

4.1.1 Entailment between BGPQs w.r.t. constraints

We start by generalizing the saturation of a query w.r.t. extra-ontological constraints. The saturated query comprises all the triples in the saturation of its body augmented with the constraints, except those triples that only follow from the ontological constraints, i.e., which are not related to what the query is asking for.

Definition 3 (Saturation w.r.t. constraints). Let \( \mathcal{R} \) be a set of RDF entailment rules, \( \mathcal{O} \) a set of RDFS statements, and \( q \) a BGPQ. The saturation of \( q \) w.r.t. \( \mathcal{O} \), denoted \( q^\mathcal{O} \), is a BGPQ with the same answer variables as \( q \) and whose body, denoted \( \text{body}(q^\mathcal{O}) \), is the maximal subset of \( \mathcal{O} \cup \text{body}(q) \) such that for any of its subset \( \mathcal{S} \) if \( \mathcal{O} \models \mathcal{S} \) holds then \( \text{body}(q) \models \mathcal{S} \) holds.

The above definition precisely characterizes \( \text{body}(q^\mathcal{O}) \) as the non-hatched subset of \( \mathcal{O} \cup \text{body}(q) \) shown in Figure 5. Also, remark that this definition coincides with the standard one of saturation (Section 2) when the set \( \mathcal{O} \) of ontological constraints is empty.

Figure 7 shows a set \( \mathcal{O} \) of ontological constraints, as well as the saturations of the BGPQs \( q_1 \) and \( q_2 \) in Figure 6 w.r.t. \( \mathcal{O} \), named \( q_1^\mathcal{O} \) and \( q_2^\mathcal{O} \) respectively.

The theorem below states the fundamental properties for a query and its saturation w.r.t. ontological constraints: they are equivalent for the central RDF reasoning tasks of query entailment and query answering.

Theorem 4. Let \( \mathcal{R} \) be a set of RDF entailment rules, \( \mathcal{O} \) a set of RDFS statements, and \( q \) a BGPQ whose saturation w.r.t. \( \mathcal{O} \) is \( q^\mathcal{O} \). For any RDF graph \( \mathcal{G} \) whose set of RDFS statements is \( \mathcal{O} \),

1. \( \mathcal{G} \models q \) holds iff \( \mathcal{G} \models q^\mathcal{O} \) holds,
2. \( q(\mathcal{G}) = q^\mathcal{O}(\mathcal{G}) \) holds.

Proof. Let us first show item 1). Observe that \( \mathcal{G} \models q \) iff \( \mathcal{G} \models q^\mathcal{O} \) holds iff \( \mathcal{G}^\mathcal{O} \models q \) iff \( \mathcal{G}^\mathcal{O} \models q^\mathcal{O} \) holds (recall Section 2). If \( \mathcal{G}^\mathcal{O} \models q^\mathcal{O} \) holds for some homomorphism \( \phi \), then \( \mathcal{G}^\mathcal{O} \models q \) holds also since \( \text{body}(q) \subseteq \text{body}(q^\mathcal{O}) \). Now, if \( \mathcal{G}^\mathcal{O} \models q \) holds for some homomorphism \( \phi \), consider the subset \( \text{body}(q) \) of \( \mathcal{G}^\mathcal{O} \). Since \( \mathcal{O} \subseteq \mathcal{G} \), and by definition of the saturation of \( q \) w.r.t. \( \mathcal{O} \), \( \phi \) can obviously be extended to a homomorphism \( \phi' \) such that \( [\text{body}(q^\mathcal{O})]_{\phi'} \subseteq \mathcal{G}^\mathcal{O} \), which maps \( \text{body}(q^\mathcal{O}) \) to the maximal subset of \( \mathcal{O} \cup \text{body}(q^\mathcal{O}) \subseteq \mathcal{G}^\mathcal{O} \), such that for any of its subset \( \mathcal{G}' \) if \( \mathcal{O} \models \mathcal{G}' \) then \( \text{body}(q) \models \mathcal{G}' \).

Item 2) directly follows from: (i) the fact that, above, item 1) holds for any homomorphism \( \phi \) and (ii) \( q \) and \( q^\mathcal{O} \) have, by definition, the same answer variables.

Based on the above result, we are now ready to extend the entailment relation between queries, i.e., the standard RDF generalization/specialization relation between queries, to that of entailment between queries w.r.t. ontological constraints.

Definition 4 (Entailment w.r.t. constraints). Given a set \( \mathcal{R} \) of RDF entailment rules, a set \( \mathcal{O} \) of RDFS statements, and two BGPQs \( q \) and \( q' \) with the same arity, \( q \) entails \( q' \) w.r.t. \( \mathcal{O} \), denoted \( q \models_{\mathcal{R},\mathcal{O}} q' \), iff \( q^\mathcal{O} \models q' \) holds.

The above definition coincides with the standard one of entailment between BGPQs (Section 2) when the set \( \mathcal{O} \) of ontological constraints is empty.

The BGPQ \( q(x) \leftarrow (x, r, \tau, \text{Publication}), (x, \text{hasAuthor}, y) \) is not entailed by the queries \( q_1 \) and \( q_2 \) shown in Figure 6, while it is entailed by these queries w.r.t. the set \( \mathcal{O} \) of ontological constraints displayed in Figure 7: \( q_2^\mathcal{O} \models q' \) (resp. \( q_2^\mathcal{O} \models q'' \)) holds for \( \phi_1(x) = x_1 \) and \( \phi_1(y) = y_1 \) (resp. \( \phi_2(x) = x_2 \) and \( \phi_2(y) = y_2 \)).

The next theorem establishes the fundamental properties for a query entailed by another w.r.t. ontological constraints: the former generalizes the latter for the central RDF reasoning tasks of query entailment and query answering.

Theorem 5. Let \( \mathcal{R} \) be a set of RDF entailment rules, \( \mathcal{O} \) a set of RDFS statements, and two BGPQs \( q \) and \( q' \) such that \( q \models_{\mathcal{R},\mathcal{O}} q' \). For any RDF graph \( \mathcal{G} \) whose set of RDFS statements is \( \mathcal{O} \),

1. if \( \mathcal{G} \models q \) holds then \( \mathcal{G} \models q' \) holds,
2. \( q(\mathcal{G}) \subseteq q'(\mathcal{G}) \) holds.

Proof. Let us first show item 1). By Theorem 4, item 1), \( \mathcal{G} \models q \) holds iff \( \mathcal{G} \models q^\mathcal{O} \) holds. Consider some \( \phi \) such that \( \mathcal{G} \models \mathcal{G}^\mathcal{O} \models q^\mathcal{O} \) holds. Also, by Definition 4, consider some \( \phi' \) such that \( q^\mathcal{O} \models q' \). By composing \( \phi \) and \( \phi' \) into \( \psi \), we get \( \mathcal{G} \models q' \), i.e., \( \mathcal{G} \models q' \).

Item 2) directly follows from: (i) the fact that, above, item 1) holds for any pair of homomorphisms \( \phi, \phi' \) such that \( \mathcal{G} \models q^\mathcal{O} \models q'' \) and \( q^\mathcal{O} \models q' \) hold, and (ii) \( q \) and \( q^\mathcal{O} \) have, by definition, the same answer variables.

Finally, it is worth noting that our notion of entailment between BGPQs w.r.t. ontological constraints is the RDF counterpart to that of query entailment under deductive constraints in a database setting [16].

4.1.2 1gg of BGPQs w.r.t. constraints

With the above new RDF generalization/specialization relation between queries endowed with background knowledge, we are now ready to define the 1gg of queries w.r.t. ontological constraints.
Definition 5 (lgg of BGPQs). Let \( q_1, \ldots, q_n \) be BGPQs with the same arity, \( R \) a set of RDF entailment rules and \( \mathcal{O} \) a set of RDFS statements.

- A generalization of \( q_1, \ldots, q_n \) w.r.t. \( \mathcal{O} \) is a BGPQ \( q_0 \) such that \( q_i \models_{\mathcal{R}, \mathcal{O}} q_0 \) holds for \( 1 \leq i \leq n \).

- A least general generalization of \( q_1, \ldots, q_n \) w.r.t. \( \mathcal{O} \) is a generalization \( q_{\text{lgg}} \) of \( q_1, \ldots, q_n \) w.r.t. \( \mathcal{O} \) such that for any other generalization \( q_0 \) of \( q_1, \ldots, q_n \) w.r.t. \( \mathcal{O} \), \( q_{\text{lgg}} \models_{\mathcal{R}, \mathcal{O}} q_0 \) holds.

When \( \mathcal{O} \) is empty, the above definition is the direct adaptation of that of \( \text{lgg} \) of RDF graphs to the case of BGPQs, using standard entailment between BGPQs.

Importantly, in our RDF and SPARQL setting, the following holds:

Theorem 6. An \( \text{lgg} \) of BGPQs w.r.t. RDFS statements may not exist for some set of RDF entailment rules; when it exists, it is unique up to entailment (\( \models_{\mathcal{R}, \mathcal{O}} \)).

Proof. It is easy to show that an \( \text{lgg} \) of BGPQs may not exist. For instance, consider the BGPQs \( q_1(x_1) \leftarrow (x_1, \text{hasAuthor}, y_1) \) asking for the resources having some author, and \( q_2(x_2) \leftarrow (y_2, \text{hasAuthor}, x_2) \) asking for the authors of some resource. Clearly, when the set \( \mathcal{R} \) of entailment rules is empty or comprises the rules in Table 2, no BGPQ can generalize \( q_1 \) and \( q_2 \); hence there is no \( \text{lgg} \) of them. By contrast, if we use the complete set of RDF entailment rules, an \( \text{lgg} \) of \( q_1 \) and \( q_2 \) is \( \text{lgg}(x) \leftarrow (x, \tau, \text{rdf:Resource}) \), since every RDF value is an instance of the built-in class rdf:Resource.

When an \( \text{lgg} \) of BGPQs w.r.t. RDFS constraints exists, it is unique up to entailment, i.e., is semantically unique, because \( q_{\text{lgg}} \models_{\mathcal{R}, \mathcal{O}} q_0 \) holds for any \( q_0 \) in Definition 5. If it were that queries have multiples \( \text{lgg} \) incomparable w.r.t. entailment, say the BGPQs \( \text{lgg}_1(x), \ldots, \text{lgg}_m(x) \), the BGPQ defined as \( q_{\text{lgg}}(x) \leftarrow \text{body}(\text{lgg}_1) \cup \cdots \cup \text{body}(\text{lgg}_m) \) would be a single strictly more specific \( \text{lgg} \), a contradiction.

Though unique up to entailment, there exist many syntactic variants (an infinity actually) of an \( \text{lgg} \) due to redundant triples, i.e., triples entailed by others within the \( \text{lgg} \). For example, think of an \( \text{lgg} \) \( q_{\text{lgg}}(x) \leftarrow (x, \tau, A), (x, \tau, B), (x, y, z) \) w.r.t. the set of constraints \( \mathcal{O} = \{(A, \lessdot_{\mathcal{X}} B), (B, \lessdot_{\mathcal{X}} A)\} \), which asks for resources of types \( A \) and \( B \) that are somehow related to some resource, and it is known that \( A \) and \( B \) are equivalent classes. Clearly, different equivalent and minimal variants (w.r.t. the number of triples) of this \( \text{lgg} \) are \( q_{\text{lgg}}(x) \leftarrow (x, \tau, A) \) and \( q_{\text{lgg}}(x) \leftarrow (x, \tau, B) \), since \( (x, y, z) \) is entailed by each of the two other triples, and \( (x, \tau, B) \) is entailed by \( (x, \tau, A) \) w.r.t. \( \mathcal{O} \), and vice versa, because \( A \) and \( B \) are equivalent. Importantly, redundancy of triples is not specific to \( \text{lggs} \) of BGPQs w.r.t. RDFS constraints, since obviously any BGPQ may feature redundancy. The detection and elimination of such redundancy have been studied in the literature, hence we focus in this work on learning some \( \text{lgg} \) of BGPQs w.r.t. RDFS constraints; learning as minimal as possible \( \text{lgg} \) is a perspective of this work discussed in Section 8.

The proposition below is the counterpart of Proposition 1 for RDF graphs. It shows that an \( \text{lgg} \) of \( n \) BGPQs, with \( n \geq 3 \), can be defined (hence computed) as a sequence of \( n - 1 \) \( \text{lggs} \) of two BGPQs. That is, assuming that \( \ell_{k \geq 2} \) is an operator computing an \( \text{lgg} \) of \( k \) input BGPQs, the next proposition establishes that:

\[
\ell_3(q_1, q_2, q_3) \equiv_{\mathcal{R}, \mathcal{O}} \ell_2(\ell_2(q_1, q_2), q_3)
\]

\[
\ell_n(q_1, \ldots, q_n) \equiv_{\mathcal{R}, \mathcal{O}} \ell_2(\ell_2(\ell_2(q_1, q_2), q_3) \cdots q_{n-1}, q_n)
\]

Proposition 3. Let \( q_1, \ldots, q_{n \geq 3} \) be \( n \) BGPQs, \( \mathcal{O} \) a set of RDFS statements and \( \mathcal{R} \) a set of RDF entailment rules.
exists the BGPQ \( q_{1:2} \) w.r.t. \( q \) holds and (ii) for any BGPQ \( q' \) such that \( q_{1:2} \leq q \), \( q_{1:2} \) holds, \( q_{1:2} = \bot \).

Let us show that the above lemma holds.

Suppose that \( q_{1:2} \) is an \( \text{lgg} \) of \( q_{1,...,q_{n-1}} \) w.r.t. \( \mathcal{O} \), i.e., by Definition 5: \( (i) \) \( q_{1:2} \leq q \) holds and (ii) for any BGPQ \( q' \) such that \( q_{1:2} \leq q \), \( q_{1:2} \) holds, \( q_{1:2} = \bot \).

Clearly, \( q_{1:2} \) is a possible value for \( q \) above, and when \( q_{1:2} \leq q \) implies \( q_{1:2} = \bot \), hence \( q_{1:2} = \bot \) must hold. \( \Box \)

Let us prove Proposition 3 using the above lemma.

\[ \Rightarrow \text{ Assume that } q_{1:2} \text{ is an } \text{lgg} \text{ of } q_{1,...,q_{n-1}} \text{ w.r.t. } \mathcal{O} \text{ and let us show that it is also an } \text{lgg} \text{ w.r.t. } \mathcal{O} \text{ of some } q_{1:2} \text{ of } q_{1,...,q_{n-1}} \text{ and } q_{n} \text{ w.r.t. } \mathcal{O}. \]

By Definition 5, because \( q_{1:2} \) is an \( \text{lgg} \) of \( q_{1,...,q_{n}} \) w.r.t. \( \mathcal{O} \), we have: \( (i) q_{1:2} \leq q \) holds and (ii) for any BGPQ \( q' \) such that \( q_{1:2} \leq q \), \( q_{1:2} = \bot \)

\[ = \text{ Assume that } q_{1:2} \text{ is an } \text{lgg} \text{ w.r.t. } \mathcal{O} \text{ of an } \text{lgg} \text{ of } q_{1,...,q_{n-1}} \text{ and } q_{n} \text{ w.r.t. } \mathcal{O}. \]

\[ \triangleleft \text{ Assume that } q_{1:2} \text{ is an } \text{lgg} \text{ w.r.t. } \mathcal{O} \text{ of an } \text{lgg} \text{ of } q_{1,...,q_{n-1}} \text{ and } q_{n} \text{ w.r.t. } \mathcal{O}. \]

\[ \Rightarrow \text{ Assume that } q_{1:2} \text{ is an } \text{lgg} \text{ w.r.t. } \mathcal{O} \text{ of an } \text{lgg} \text{ of } q_{1,...,q_{n-1}} \text{ and } q_{n} \text{ w.r.t. } \mathcal{O}. \]

By Definition 5, \( (i) q' \leq q \) holds and \( q' \leq q \) holds, hence \( q_{1:2} \leq q \).

\[ \triangleleft \text{ Assume that } q_{1:2} \text{ is an } \text{lgg} \text{ w.r.t. } \mathcal{O} \text{ of an } \text{lgg} \text{ of } q_{1,...,q_{n-1}} \text{ and } q_{n} \text{ w.r.t. } \mathcal{O}. \]

\[ \Rightarrow \text{ Assume that } q_{1:2} \text{ is an } \text{lgg} \text{ w.r.t. } \mathcal{O} \text{ of an } \text{lgg} \text{ of } q_{1,...,q_{n-1}} \text{ and } q_{n} \text{ w.r.t. } \mathcal{O}. \]

Based on the above result, without loss of generality, we focus in the next section on the following problem:

**Problem 2.** Given two BGPQs \( q_1, q_2 \) with same arity, a set \( \mathcal{O} \) of RDFs statements, and a set \( \mathcal{R} \) of RDF entailment rules, we want to compute some \( \text{lgg} \) of \( q_1 \) and \( q_2 \) w.r.t. \( \mathcal{O}. \)

### 4.2 Computing an \( \text{lgg} \) of BGPQs

We first devise the notion of cover query of two BGPQs \( q_1 \) and \( q_2 \) (to be defined shortly, Definition 6 below) and we show (Theorem 7) that it corresponds to an \( \text{lgg} \) of \( q_1 \).

In particular, \( q_1 \) and \( q_2 \) in case extra ontological constraints and RDF entailment are ignored. Then, we show (Theorem 8) that their enhanced \( \text{lgg} \) as defined in Definition 5, i.e., which does consider extra ontological constraints and RDF entailment, can be obtained as the cover query of the saturations of \( q_1 \) and of \( q_2 \) with the ontological constraints and RDF entailment rules.

**Definition 6 (Cover query).** Let \( q_1, q_2 \) be two BGPQs with the same arity \( n \).

If there exists the BGPQ \( q \) such that

- \( \{t_1,t_2,t_3\} \subseteq \text{body}(q_1) \) and \( \{t_4,t_5,t_6\} \subseteq \text{body}(q_2) \) iff \( \{t_7,t_8,t_9\} \subseteq \text{body}(q) \) with, for \( 1 \leq i \leq 3, t_{i+3} = t_i \), if \( t_i \in U \cup L \), otherwise \( t_{i+3} \in \text{the variable} v_{t_i,t_{i+3}} \).

\[ \text{head}(q) = q(x_1^1,...,x_n^1), \text{head}(q_1) = q(x_1^1,...,x_n^1) \]

then \( q \) is the cover query of \( q_1 \) and \( q_2 \).

The cover query body is defined (first item above) as a slight generalization of the cover graph of two RDF graphs (Definition 1 and Theorem 2), so that it corresponds to an \( \text{lgg} \) of the bodies of \( q_1 \) and \( q_2 \) when both extra ontological constraints and RDF entailment are ignored. Recall that the bodies of the BGPQs are BGPQs, which generalize RDF graphs by allowing variables in triples (Section 2). In particular, unassigned values are allowed in the property position of BGP triples (using variables), whereas this was not possible in RDF graph triples in which the property values must be URIs. To generalize the results of Theorem 2 from RDF graphs to BGPQs, \( (i) \) variables are used instead of blank nodes to generalize distinct values in the least general anti-unifications of triples begetting the cover query body (as they have the same semantics within a BGP, but only variables can be used at subject, object and property positions in triples), \( (ii) \) the cover query body comprises the least general anti-unifications of every \( q_1 \) triple with every \( q_2 \) triple instead of just those pairs of triples with same property URI (as now distinct property values can be generalized using variables), and \( (iii) \) again, across these anti-unifications characterizing the body of \( q \), we consistently name the variables generalizing same pair of the distinct \( q_1 \) and \( q_2 \) values (so as to capture the common structure between the bodies of \( q_1 \) and \( q_2 \)).

The cover query head is simply defined (second item in Definition 6) as the least general anti-unification of the \( q_1 \)'s head and \( q_2 \)'s head, in order to model the least general set of result tuples generalizing those of \( q_1 \) and of \( q_2 \).

Figure 8 (left) shows the cover query \( q \) of the BGPQs \( q_1 \) and \( q_2 \) displayed in Figure 6. The head \( q' = (v_{x_1^1},...) \) results from anti-unifying \( q_1(x_1) \) and \( q_2(x_2) \); its body consists of the triple \( (v_{x_1^1},\tau,\text{conf}) \), which results from anti-unifying \((x_1,\tau,\text{conf}) \), in \( q_1 \) and \((x_2,\tau,\text{conf}) \), in \( q_2 \), of the triple \( (v_{x_1^1},\tau,\text{conf}) \), which results from anti-unifying \((x_1,\tau,\text{conf}) \), in \( q_1 \) and \((x_2,\tau,\text{conf}) \), in \( q_2 \), etc.

Importantly, a cover query of two BGPQs may not exist \( \{...\} \) in Definition 6. For instance, recall the BGPQs \( q_1 \) and \( q_2 \) previously introduced in subsection 4.1.2, with which we pointed out that an \( \text{lgg} \) may no exists. Their cover query does not exist either, since Definition 6 leads to \( q(x_1^1,...,x_n^1) \), which is not a BGPQ...
$(v_{x_1} x_2$ does not appear in $q$‘s body).

The next theorem formalizes the above discussion:

**Theorem 7.** Given two BGPQs $q_1, q_2$ with the same arity and empty sets $R$ of RDF entailment rules and $\mathcal{O}$ of RDFS statements:

1. the cover query of $q_1$ and $q_2$ exists iff an $\mathcal{Lg}$ of $q_1$ and $q_2$ exists;
2. the cover query of $q_1$ and $q_2$ is an $\mathcal{Lg}$ of $q_1$ and $q_2$.

**Proof.** The proof can be directly derived from that of the more general Theorem 8 (see below), noting that in the particular case of Theorem 7 $R = \emptyset$ and $\mathcal{O} = \emptyset$ hold, hence $q_1^\infty = q_1$ and $q_2^\infty = q_2$ hold.

In Figure 8 (left), the cover query $q'$ of the BGPQs $q_1$ and $q_2$ displayed in Figure 6 is therefore an $\mathcal{Lg}$ of them when both extra ontological constraints and RDF entailment are ignored (Theorem 7). Remark that $q'$ is equivalent to the minimal $\mathcal{Lg}$ of $q_1$ and $q_2$ shown Figure 6: $q_{\mathcal{Lg}}$.

We provide below the worst-case time to compute a cover query, and its size.

**Proposition 4.** The cover query of two BGPQs $q_1$ and $q_2$ can be computed in $O(|\text{body}(q_1)| \times |\text{body}(q_2)|)$; its size is $|\text{body}(q_1)| \times |\text{body}(q_2)|$.

The next theorem generalizes the preceding one in order to use the notion of cover query to compute an $\mathcal{Lg}$ of two queries w.r.t. extra ontological constraints and any set of RDF entailment rules.

**Theorem 8.** Given a set $R$ of RDF entailment rules, a set $\mathcal{O}$ of RDFS statements and two BGPQs $q_1, q_2$ with the same arity,

1. the cover query of $q_1^\infty$ and $q_2^\infty$ exists iff an $\mathcal{Lg}$ of $q_1$ and $q_2$ w.r.t. $\mathcal{O}$ exists;
2. the cover query of $q_1^\infty$ and $q_2^\infty$ is an $\mathcal{Lg}$ of $q_1$ and $q_2$ w.r.t. $\mathcal{O}$.

**Proof.** We start by showing that the cover query $q$, when it exists, is a generalization w.r.t. $\mathcal{O}$ of $q_1$ and $q_2$, i.e., $q_1 \models_{\mathcal{O}} q$ and that $q_2 \models_{\mathcal{O}} q$. Clearly, once proved, this entails that if $q$ exists, then an $\mathcal{Lg}$ of $q_1$ and $q_2$ w.r.t. $\mathcal{O}$ exists.

Consider the BGPQ $q'$ obtained from $q$ by replacing every variable $v_{x_1} x_2$ by the value $t_1$. Clearly, $q' \models_{\mathcal{O}} q$ and, by construction of $q$, $q' = q_1$, i.e., $q_1 \models_{\mathcal{O}} q$. Showing that $q_2 \models_{\mathcal{O}} q$ holds is done in a similar way.

Therefore, the cover query $q$ is a generalization of $q_1$ and $q_2$ w.r.t. $\mathcal{O}$, thus an $\mathcal{Lg}$ of $q_1$ and $q_2$ w.r.t. $\mathcal{O}$ also exists.

Now, let us show that the cover query $q$, when it exists, is an $\mathcal{Lg}$ of $q_1$ and $q_2$ w.r.t. $\mathcal{O}$. Let $q_{\mathcal{Lg}}$ be any $\mathcal{Lg}$ of $q_1$ and $q_2$ w.r.t. $\mathcal{O}$. Without loss of generality, assume that BGPQs do not contain blank nodes (recall that they can be equivalently replaced by non-answer variables, Section 2). Assume that $q_1, q_2, q_{\mathcal{Lg}}$ heads are $q(x), q_1(x_1), q_2(x_2), q_{\mathcal{Lg}}(x_{\mathcal{Lg}})$ respectively.

To prove our claim, we need to show that $q \models_{\mathcal{O}} q_{\mathcal{Lg}}$ holds. Since $q_{\mathcal{Lg}}$ is an $\mathcal{Lg}$ of $q_1$ and $q_2$ w.r.t. $\mathcal{O}$, there exist two homomorphisms $\phi_1$ and $\phi_2$ from the variables in $q_{\mathcal{Lg}}$ to the values (variables and constants) in $q_1^\infty$ and in $q_2^\infty$ respectively, such that $\text{body}(q_{\mathcal{Lg}}) \subseteq \text{body}(q_1^\infty)$ and $\phi_1(x_{\mathcal{Lg}}) = x_1$, and similarly, $\text{body}(q_{\mathcal{Lg}}) \subseteq \text{body}(q_2^\infty)$ and $\phi_2(x_{\mathcal{Lg}}) = x_2$. Consider the BGPQ $q'_{\mathcal{Lg}}$ obtained from $q_{\mathcal{Lg}}$ by replacing every variable $v$ by $v_{x_1} x_2$ where $t_1 = \phi_1(v)$ and $t_2 = \phi_2(v)$. Clearly, $q_{\mathcal{Lg}} \equiv_{\mathcal{O}} q'_{\mathcal{Lg}}$ holds. Assume that the head of $q'_{\mathcal{Lg}}$ is $q'_{\mathcal{Lg}}(x_{\mathcal{Lg}})$; by construction is has the same head as $q$. Let us show that $q \models_{\mathcal{O}} q'_{\mathcal{Lg}}$ holds, i.e., there exists a homomorphism $\phi$ from the variables in $q'_{\mathcal{Lg}}$ to values in $q_2^\infty$ such that $\text{body}(q'_{\mathcal{Lg}}) \subseteq \text{body}(q_2^\infty)$ and $\phi(x_{\mathcal{Lg}}) = x$. By definition of $q$, it is easy to see that the above holds when $\phi$ maps $v_{x_1} x_2$ either to $t_1$ if $t_1 = t_2$ and $t_2 \in U \cup \mathcal{L}$, or to itself otherwise. Importantly, this entails (i) the existence of the cover query $q$ when an $\mathcal{Lg}$ exists and (ii) that $q \models_{\mathcal{O}} q_{\mathcal{Lg}}$ holds, hence $q \models_{\mathcal{O}} q_{\mathcal{Lg}}$ holds, and therefore $q$ is an $\mathcal{Lg}$ of $q_1$ and $q_2$ w.r.t. $\mathcal{O}$.

As an immediate consequence of the above results, we get the following worst-case time to compute an $\mathcal{Lg}$ of two BGPQs $q_1$ and $q_2$, and its size. We assume given the saturation $q_1^\infty$ and $q_2^\infty$ w.r.t. the sets $\mathcal{O}$ of ontological constraints and $R$ of RDF entailment rules under consideration, as the times to compute $q_1^\infty$ and $q_2^\infty$, and their sizes, depend on the particular $\mathcal{O}$ and $R$ at hand.

**Corollary 2.** An $\mathcal{Lg}$ of two BGPQs $q_1$ and $q_2$ can be computed in $O(|\text{body}(q_1^\infty)| \times |\text{body}(q_2^\infty)|)$ and its size is $|\text{body}(q_1^\infty)| \times |\text{body}(q_2^\infty)|$.

Similarly to the preceding section, computing naively the cover query-based $\mathcal{Lg}$ of $n$ BGPQs of size $M$ based on Proposition 3 leads to an $\mathcal{Lg}$ of size $M^n$. However, remov-
ing the redundant triples from the intermediate and final cover query-based 1gggs limits their size to at most $M$.

Figure 8 (right) displays the cover query of the BGPQs $q_1 \circ G$ and $q_2 \circ G$ shown in Figure 7. It is therefore (Theorem 8) an 1gg of the BGPQs $q_1$ and $q_2$ shown in Figure 6 w.r.t. the ontological constraints shown in Figure 7, using the RDF entailment rules shown in Table 2.

Figure 8 exemplifies the benefits of taking account extra ontological constraints modeling background knowledge when identifying the commonalities between queries, thus of endowing the RDF relation of generalization/specialization between queries with such knowledge. When background knowledge is ignored (left), we only learn that both $q_1$ and $q_2$ ask for the resources having some type. In contrast, when we do consider background knowledge (right), we further learn that these resources, which both $q_1$ and $q_2$ ask for, are publications, which have some researcher as author.

5. ALGORITHMS

We provide algorithms to compute 1gggs of RDF graphs and BGPQs, based on the results obtained in the preceding Sections. In Section 5.1, we present an algorithm for computing the least general anti-unifiers of tuples of RDF values (URIs, literals, blank nodes and variables), i.e., of triples, triple patterns or query heads, on the definitions of cover graphs and queries rely. Then, in Section 5.2, we give three algorithms to compute a cover graph-based 1gg of RDF graphs when the input and output RDF graphs fit in memory, in data management systems or in MapReduce clusters. Finally, in Section 5.3, we provide an algorithm to compute a cover query-based 1gg of BGPQs.

5.1 Least general anti-unification

Algorithm 1, called 1gau, computes a least general anti-unifier $(t'_1, \ldots, t'_n)$ of two $n$-ary tuples of RDF values $(t_1, \ldots, t_n)$ and $(t'_1, \ldots, t'_n)$, made of constants (URIs and literals), blank nodes and variables. This is achieved by setting the $i$th value $t'_i$ of the output tuple to the least general generalization of the values found at the $i$th positions of the two input tuples: $t_i$ and $t'_i$. Recall that a pair of a constant is generalized by that constant itself, while in all other cases the generalization is either a blank node in case of RDF graph triple values (Section 3.2) or a variable in case of query head answer variables or of query body triple pattern values (Section 4.2). Generalizing pairs of different values with blank nodes or variables is controlled with the Boolean bnodes parameter. Crucially, these generated blank nodes or variables adopt the consistent naming scheme devised in Sections 3.2 and 4.2 that allows us preserving the common input RDF graphs (resp. BGPQs) structure across the anti-unifications of triples (resp. triple patterns).

5.2 1gggs of RDF graphs

Following Definition 2, Algorithm 2, called 1gg4g, computes the cover graph $G$ of two input RDF graphs $G_1$ and $G_2$: $G$ comprises the least general anti-unification of every pair of $G_1$ and $G_2$ triples with same property. Therefore, given two RDF graphs $G_1$ and $G_2$, a call 1gg4g($G_1$, $G_2$) produces the cover graph-based 1gg of $G_1$ and $G_2$ ignoring RDF entailment (Theorem 2), while a call 1gg4g($G_1 \circ G$, $G_2 \circ G$) produces the cover graph-based 1gg of $G_1$ and $G_2$ taking into account the set of RDF entailment rules at hand (Theorem 3). In the latter case, the input RDF graphs can be saturated using standard algorithms implemented in RDF data management systems, like Jena [3] and Virtuoso [8].

Importantly, 1gg4g assumes that the input RDF graphs, as well as their output cover graph, fit in memory. Checking whether this is the case for the input RDF graphs under consideration can be done as follows.

The size of the input RDF graphs $G_1$ and $G_2$ can be computed with the following SPARQL queries counting how many triples it holds per distinct property

```
SELECT SUM(?) as n
FROM 1gg4g(G1, G2)
GROUP BY ?p

This can be computed with the SPARQL query:

SELECT COUNT(*) WHERE 1gg4g(G1, G2)

Then, since every $G_i$ triple with property $p$ anti-unifies with every $G_j$ triple with same property $p$, in order to beget $G$, the size of $G$ is:

```
|G| = \sum_{(p,n_1)\in S_{G_1} \cap (p,n_2)\in S_{G_2}} n_1 \times n_2
```

Overall, $|G|$ can be computed with the SPARQL query:

```
SELECT COUNT(*) WHERE 1gg4g(G1, G2)
```

When the input RDF graphs or their output cover graph cannot fit in memory, we propose variants of 1gg4g that either assume that RDF graphs are stored in data manage-

Algorithm 1 Least general anti-unification: 1gau

**In:** $T_1 = (t_1, \ldots, t_n)$ and $T_2 = (t'_1, \ldots, t'_n)$, boolean bnodes

**Out:** least general anti-unification $T$ of $T_1$ and $T_2$

1: for $i = 1$ to $n$ do
2: if $t_i = t'_i$ and $t_i \in \mathcal{U} \cup \mathcal{L}$ then
3: $t'_i \leftarrow t_i$
4: else if $t'_i$ is a node then
5: $t'_i \leftarrow \text{generalization of a constant by itself}$
6: else
7: $t'_i \leftarrow \text{generalization by blank node}$
8: end if
9: end for
10: return $(t'_1, \ldots, t'_n)$

Algorithm 2 Cover graph of two RDF graphs: 1gg4g

**In:** RDF graphs $G_1$ and $G_2$

**Out:** $G$ is the cover graph of $G_1$ and $G_2$

1: $G \leftarrow \emptyset$
2: for all $T_1 = (s_1, p_1, o_1) \in G_1$ do
3: for all $T_2 = (s_2, p_2, o_2) \in G_2$ with $p_1 = p_2$ do
4: $G \leftarrow G \cup \{1gau(T_1, T_2, true)\}$
5: end for
6: end for
7: return $G$
Algorithm 3: Cover graph of two RDF graphs: $1gg4g$-dms
\begin{verbatim}
In: \hspace{2em} \text{cursor } c_1 \text{ on RDF graph } \mathcal{G}_1, \text{cursor } c_2 \text{ on RDF graph } \mathcal{G}_2, \text{write access to an empty RDF graph } \mathcal{G}, \text{integer } n
\end{verbatim}
\begin{align*}
\text{Out: } \mathcal{G} \text{ is the cover graph of } \mathcal{G}_1 \text{ and } \mathcal{G}_2 \text{.}
\end{align*}
\begin{algorithm}
\begin{algorithmic}[1]
\State $\text{c}_1.\text{init}() \triangleright \text{c}_1 \text{ at beginning of } \mathcal{G}_1$
\While {$B_1 = \text{c}_1.\text{next}(n)$} \Comment {fetch next $n$ triples}
\State $\text{c}_2.\text{init}() \triangleright \text{c}_2 \text{ at beginning of } \mathcal{G}_2$
\EndWhile
\ForAll {$B_2 = \text{c}_2.\text{next}(n)$} \Comment {fetch next $n$ triples}
\EndFor
\ForAll {$T_2 = (s_2, p_2, o_2) \in B_2$ with $p_1 = p_2$} \Comment {for all triples with the same property}
\State \text{insert } \text{lgg}_4(\text{lgg}_4(T_1, T_2, \text{true})) \text{ into } \mathcal{G}$
\EndFor
\EndWhile
\end{algorithmic}
\end{algorithm}

5.2.2 Handling huge RDF graphs using MapReduce

Algorithm 4, called $1gg4g$-mr, is a MapReduce (MR) variant of $1gg4g$. MR is a popular massively parallel programming framework [21], implemented by many large-scale data processing systems, like Hadoop [2] and Spark [7], which orchestrate clusters of compute nodes.

A MR program is organized in successive jobs, each of which comprises a Map task followed by a Reduce task. The Map task consists in reading some input data from the distributed file system\(^2\) of the cluster, so as to partition the data into $(x, v)$ key-value pairs. Importantly, an MR engine transparently processes the Map task by running Mapper processes in parallel on cluster nodes, each process taking care of partitioning a portion of the input data by applying a Map(key: file, value: data unit) function on every data unit of a given input file. Key-value pairs thus produced are shuffled across the network, so that all pairs with same key $(x, v_1) \ddots (x, v_n)$ are shipped to a same compute node. The Reduce task then consists in running Reducer processes in parallel, for every distinct key $k$ received by every compute node. Each process takes care of the set $V$ of values $\{v_1, \ldots, v_n\}$ emitted with key $k$, by applying a Reduce(key: $k$, values: $V$) function, and writing its results in a file. The result of an MR job, comprises the data, stored in a distributed fashion, in all the files output by Reducers.

In $1gg4g$-mr, the Map function applies to every $(s_1, p, o_1)$ triple of the input RDF graph $\mathcal{G}_1$, stored in file $\mathcal{G}_1$, and produces the corresponding key-value pair $(\text{lgg}_4(T_1, (s_1, p, o_1)))$, for $i \in \{1, 2\}$. Hence, all the $G_1$ and $G_2$ triples with a same key/property $p$ are shipped to the same cluster node. Then, similarly to $1gg4g$ at lines 2-6, the Reduce functions process, on each node, the set $V$ of values emitted for every received key $p$. The least general anti-unification triples obtained at line 4 are stored in the output file $G$-$p$. At the end of the MR job, the $1gg$ $G$ of $G_1$ and $G_2$ is stored in the $G$-$+$ files of the distributed file system, where $+$ denotes any key/property $p$.

In $1gg4g$-mr, the Map function holds at most a single $G_1$ or $G_2$ triple in memory. In contrast, the worst-case number of triples handled by the Reduce function for a given key $p$ is: $M = |G_1| + |G_2| + |G_1 \times G_2|$. This upper bound is met in the unlikely case where $G_1$ and $G_2$ use the same property $p$ in all their triples. Similarly to $1gg4g$-dms, this upper bound can set to $M = 2^n + 1$, with $3 \leq |G_1| + |G_2| + |G_1 \times G_2|$, and then by processing every pair of n-triples of $G_1$ and $G_2$ files with an MR job (i.e., with $k_1 \times k_2$ jobs), instead of a single MR job for the entire two

\(^2\)For simplicity, we assume that input and output data of an MR job is stored on disk, like in Hadoop, while it may also reside in in-memory shared data structures, like in Spark.
input RDF graphs.

Finally, to take into account RDF entailment, the input RDF graphs \( G_1 \) and \( G_2 \) can be saturated before being stored in the MR cluster, by using standard (centralized) techniques, or within the MR cluster, by using MR-based saturation techniques [45, 44]. Also, it is worth noting that RDF graphs, thus \( \text{lgg} \)'s of RDF graphs, stored in an MR cluster can be queried with MR-based SPARQL engines [27, 30, 35, 23].

5.3 \( \text{lgg} \) of BGPQs

Following Definition 6, Algorithm 5, called \( \text{lgg}4q \), builds and returns either the cover query \( q \) of two input BGPQs \( q_1 \) and \( q_2 \) if it exists, or the ⊥ symbol otherwise.

At lines 2-6, \( \text{lgg}4q \) computes the body of \( q \), which comprises a least general anti-unifier of every pair of \( q_1 \) body and \( q_2 \) body triples (first item in Definition 6). Then, at line 7, it computes the head of \( q \) as the least general anti-unifier of the heads of \( q_1 \) and \( q_2 \) (second item in Definition 6). Finally, \( \text{lgg}4q \) checks whether \( q \) is a well-formed BGPQ, i.e., if its answer variables occur in its body triples, otherwise there is no cover query of \( q_1 \) and \( q_2 \), and also no \( \text{lgg} \) of \( q_1 \) and \( q_2 \) (recall first item of Theorems 7 and 8).

Therefore, given two BGPQs \( q_1 \) and \( q_2 \), a call \( \text{lgg}4q(q_1, q_2) \) produces the cover query-based \( \text{lgg} \) of \( q_1 \) and \( q_2 \) ignoring RDF entailment and extra ontological constraints, whenever it exists (Theorem 7), while a call \( \text{lgg}4q(q_1, q_2, q_1, q_2) \) produces the cover query-based \( \text{lgg} \) of \( q_1 \) and \( q_2 \) taking into account the set of RDF entailment rules at hand, as well as a set of extra ontological constraints, whenever it exists (Theorem 8). In the latter case, the saturated input BGPQs can be computed using Algorithm 6, called \( \text{sat}4q \).

Following Definition 3, \( \text{sat}4q \) computes the saturation of a query \( q \) w.r.t. a set \( R \) of RDF entailment rules and a set \( O \) of RDFS constraints as a BGPQ \( q^\mathcal{S} \) with same head as \( q \) (line 1), and whose body is computed based on its characterization displayed in Figure 5: \( \text{body}(q^\mathcal{S}) = q^\mathcal{S} \cap \{q \cap (\text{body}(q) \cup O) \} \) (∗). Further, since \( \text{body}(q) \cap O \) is equivalent to \( \text{body}(q) \cap O \) (∗) (because \( \text{body}(q) \) and \( O \) are equivalent w.r.t. \( R \) to \( \text{body}(q) \cap O \) (∗) holds, \( \text{sat}4q \) uses (∗∗) instead of (∗) since, once \( \text{body}(q) \cap O \) (∗) is computed (lines 2 and 3), clearly \( \text{body}(q) \cap O \) (∗) is faster to compute than \( \text{body}(q) \cap O \) (∗), thus (∗∗) is faster to compute (line 5) than (∗).

6. EXPERIMENTS

We provide an experimental assessment of our technical contribution for computing \( \text{lgg} \) of BGPQs, which goes beyond that for RDF graphs (recall that BGPQ bodies generalize RDF graphs). We experimentally study the added-value of considering background knowledge when learning \( \text{lgg} \) of queries. As our next result shows (Proposition 5), this amounts to measuring how more specific (pregnant) is an \( \text{lgg} \) of queries that considers background knowledge than an \( \text{lgg} \) of the same queries that ignores background knowledge:

**Proposition 5.** Let \( R \) be a set of RDF entailment rules, \( O \) a set of RDFS statements, and \( q_1, q_2 \) two BGPQs with same arity. An \( \text{lgg} \) \( \text{qgg}^{\text{lgg}} \) of \( q_1, q_2 \) and an \( \text{lgg} \) \( \text{qgg}^{\text{lgg}} \) of \( q_1, q_2 \) w.r.t. \( \mathcal{O} \) (Definition 5) are such that: \( \text{qgg}^{\text{lgg}} \models R \text{qgg}^{\text{lgg}} \).

**Proof.** Since the \( \text{lgg} \) of some queries are equivalent (recall that the \( \text{lgg} \) is semantically unique): i) \( \text{qgg}^{\text{lgg}} \) is equivalent to the cover query-based \( \text{lgg} \) of the saturations with \( R \) of \( q_1 \) and of \( q_2 \) w.r.t. the empty set of RDFS constraints, and ii) \( \text{qgg}^{\text{lgg}} \) is equivalent to the cover query-based \( \text{lgg} \) of the saturations with \( R \) of \( q_1 \) and of \( q_2 \) w.r.t. \( \mathcal{O} \). Further, by definition of a cover query (Definition 6), \( q \) and \( q \) have the same heads and the body of \( q \) is a subset of that \( q \), thus (iii) \( q \models R \text{qgg}^{\text{lgg}} \). Therefore, from (i), (ii) and (iii), \( \text{qgg}^{\text{lgg}} \models R \text{qgg}^{\text{lgg}} \). □

Based on the above result, as a practical metric for measuring the semantic distance between \( \text{qgg}^{\text{lgg}} \) and \( \text{qgg}^{\text{lgg}} \) through \( \models R \), and because \( \text{lgg} \) are BGPQs, we compute the gain in precision background knowledge yields w.r.t. query answering. For an RDF graph \( G \) with set \( O \) of RDFS statements, we have:

\[
\text{gain in precision (in \%)} = 1 - \frac{|\text{qgg}^{\text{lgg}}(G) \cap \text{qgg}^{\text{lgg}}(G)|}{|\text{qgg}^{\text{lgg}}(G)|} = 1 - \frac{|\text{qgg}^{\text{lgg}}(G)|}{|\text{qgg}^{\text{lgg}}(G)|} \text{ since } \text{qgg}^{\text{lgg}}(G) \subseteq \text{qgg}^{\text{lgg}}(G).
\]

In Section 6.1, we describe our experimental settings. Then, in Section 6.2, we study the saturation of a BGPQ w.r.t. ontological constraints. Finally, in Section 6.3, we study the computation of \( \text{lgg} \) with or without considering constraints, notably the gain in precision when constraints are considered.

6.1 Settings

**Software.** We implemented our technical contributions in Java 1.8, on top of the Jena 3.0.1 RDF reasoner and of PostgreSQL 9.3.11 server, all used with default settings.

- We used Jena to compute the saturation of an RDF graph, against which queries must be evaluated to obtained their complete answer sets (Section 2); we also used Jena...
Table 3: Characteristics of our test queries (top) and of their saturations w.r.t. LUBM constraints (bottom); times are in ms.

<table>
<thead>
<tr>
<th>lgg of the LUBM queries:</th>
<th>Q₁₈Q₄</th>
<th>Q₈₉Q₈</th>
<th>Q₉₁Q₈</th>
<th>Q₉₂Q₈</th>
<th>Q₉₃Q₈</th>
<th>Q₉₄Q₈</th>
<th>Q₉₅Q₈</th>
<th>Q₉₆Q₈</th>
<th>Q₉₇Q₈</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to compute q_{lgg}</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>[q_{lgg}(G_{LUBM})]</td>
<td>30 963</td>
<td>1 048 360</td>
<td>1 048 360</td>
<td>1 048 360</td>
<td>253 443</td>
<td>1 048 360</td>
<td>6 937 472</td>
<td>57 774</td>
<td></td>
</tr>
<tr>
<td>Time to compute q_{lgg}²</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>[q_{lgg²}(G_{LUBM})]</td>
<td>14 285</td>
<td>14 285</td>
<td>156 358</td>
<td>156 358</td>
<td>185 124</td>
<td>19 053</td>
<td>156 358</td>
<td>520 365</td>
<td>34 852</td>
</tr>
<tr>
<td>Gain in precision</td>
<td>53.86</td>
<td>98.63</td>
<td>94.62</td>
<td>24.49</td>
<td>82.34</td>
<td>92.48</td>
<td>94.62</td>
<td>92.49</td>
<td>39.67</td>
</tr>
</tbody>
</table>

Table 4: Characteristics of cover query-based lgg's of test queries, w or w/o using the LUBM constraints; times are in ms.

<table>
<thead>
<tr>
<th>1gg of the LUBM queries:</th>
<th>Q₁₈Q₄</th>
<th>Q₈₉Q₈</th>
<th>Q₉₁Q₈</th>
<th>Q₉₂Q₈</th>
<th>Q₉₃Q₈</th>
<th>Q₉₄Q₈</th>
<th>Q₉₅Q₈</th>
<th>Q₉₆Q₈</th>
<th>Q₉₇Q₈</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to compute q₁gg</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>[q₁gg(G_{LUBM})]</td>
<td>30 963</td>
<td>1 048 360</td>
<td>1 048 360</td>
<td>1 048 360</td>
<td>253 443</td>
<td>1 048 360</td>
<td>6 937 472</td>
<td>57 774</td>
<td></td>
</tr>
<tr>
<td>Time to compute q₂gg</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>[q₂gg(G_{LUBM})]</td>
<td>14 285</td>
<td>14 285</td>
<td>156 358</td>
<td>156 358</td>
<td>185 124</td>
<td>19 053</td>
<td>156 358</td>
<td>520 365</td>
<td>34 852</td>
</tr>
<tr>
<td>Gain in precision</td>
<td>53.86</td>
<td>98.63</td>
<td>94.62</td>
<td>24.49</td>
<td>82.34</td>
<td>92.48</td>
<td>94.62</td>
<td>92.49</td>
<td>39.67</td>
</tr>
</tbody>
</table>

to compute the saturation $q_{\Sigma}$ of a BGPQ $q$ w.r.t. a set $\Sigma$ of RDF constraints (Definition 3, Algorithm 6): we rely on Jena’s saturation, union and difference operators to compute $q_{\Sigma}$’s body.

- We used PostgreSQL to evaluate SQLized BGPQs against a saturated RDF graph stored in a triple $(a, p, o)$ table. The table is indexed by all permutations of the $a$, $p$, $o$ columns, leading to a total of 6 indexes. This indexing choice is inspired by [34, 49], to give PostgreSQL efficient query evaluation opportunities. There exists alternative data layouts like having one two-columns table $p(a, o)$ per distinct property $p$ in the RDF graph, which stores all the subject/object triples of tuples with property $p$ [14], or the elaborate layout of [12] used in DB2 RDF. However, adopting another data layout than the triple table would have no impact on our experiments, as the reported times are not related to query evaluation.

RDF entailment rules. We used the subset of standard RDF entailment rules in Table 2, which fully allows exploiting RDFs’ ontological constraints, i.e., background knowledge.

Datasets. We conducted experiments using real DBpedia data [32] and synthetic LUBM data [26].

- LUBM dataset: we generated an RDF graph $G_{LUBM}$ comprising 884k triples before saturation, including a subset $O_{LUBM}$ of 242 RDFs constraints. The saturation of $G_{LUBM}$ comprises 1.08M triples after saturation.

- DBpedia dataset: we picked four complementary files\(^3\) to build the RDF graph $G_{DBpedia}$ comprising 41.18M triples, whose subset $O_{DBpedia}$ of 30.31k RDFs constraints represents DBpedia’s background knowledge. The saturation of $G_{DBpedia}$ comprises 78.14M triples.

Queries. We used the following queries:

- LUBM queries: we borrowed from [13] a set of 9 BGPQs. They can be found in Appendix A. Their characteristics are displayed in Table 3 (top): LUBM queries have a variety of structural aspects and numbers of answers.

- DBpedia queries: we defined 42 test BGPQs, among which we picked 8 representative ones with 2 variables. These 8 queries can be found in Appendix B. Their characteristics are displayed in Table 5 (top). Also, importantly, queries $Q_1$-$Q_4$, in Table 5 (left), are heterogeneous in the sense that they differ significantly both on their structure and the kind of information they ask for, hence use many distinct classes, properties and URI values, while $Q_5$-$Q_8$, in Table 5 (right), are homogeneous and only differ in some classes, properties and URI values.

Hardware. We used an Intel Xeon (X5550) 2.67GHz machine with 32GB RAM, using Ubuntu 14.04.3 LTS (64bits).

In the sequel, all measured times are averaged over 5 warm runs and are in milliseconds.

6.2 BGPQ saturation w.r.t. constraints

Tables 3 (bottom) and 5 (bottom) show the size of our saturated test queries and the time to compute them. Enriching our test queries using the LUBM and DBpedia constraints augments their size: from $\times 2$ for $Q_{10}$ up to $\times 5$ for $Q_{22}$ in the LUBM case; from $\times 3.16$ for $Q_2$ up to $\times 4.75$ for $Q_7$ in the DBpedia case. The query saturation time is rather fast with LUBM, 23ms on average, while it is 692ms on average with DBpedia. Saturation times have been multiplied by 30 from LUBM to DBPedia, since the latter has $\times 721$ more constraints than the former.

6.3 lgg's of BGPQs w.r.t. constraints

LUBM. Table 4 (lines 1 and 3) shows that cover query-based lgg's of test queries are always computed fast whether or not we consider the LUBM constraints: 1 to 4ms when they are ignored, and 5 to 10ms when they are considered. In the latter case, overall, it takes between 50 and 59ms to compute an lgg in the worst case (i.e., when the two saturated test queries are computed in sequence before computing their cover query). Table 4 (lines 2 and 4) shows that ignoring extra constraints significantly increases the number of answers for some lgg’s, from a small $\times 1.32$ for $Q_2$-$Q_8$ up to a striking $\times 73.39$ for $Q_2$-$Q_4$, with a significant average...
of $\times 16.45$. This translates into the precision gains shown at line 5 (Table 4): 74.80% overall, 98.63% for queries $Q_2$ and $Q_4$, and 24.49% for $Q_2$ and $Q_4$. $1gg$s with same number of answers in Table 4 were found either equivalent or almost equivalent so that their semantic difference is not visible on the LUBM dataset, e.g., the $1gg$s of $Q_2Q_3$, $Q_2Q_4$ and $Q_1Q_8$ with 1 048 360 answers (i.e., ignoring ontological constraints) are asking for all the (distinct) subject/object pairs in triples in the LUBM RDF graph; the $1gg$s of $Q_2Q_9$, $Q_2Q_3$ with 56 358 answers (i.e., considering ontological constraints) ask respectively for faculties with some degree from some university: they have the same number of answers in our LUBM RDF graph because all faculties have some doctoral degree and all employees with some doctoral degree are faculties.

$DBpedia$. First, as Table 6 (lines 1 and 3) shows, the cover query-based $1gg$s of test queries are always computed fast whether or not the DBpedia constraints are considered: from 3 to 6ms when ignored, to 13 to 18ms when considered. In the latter case, overall, it takes between 1.322 and 1.457s to compute an $1gg$ in the worst case, i.e., when the two saturated test queries are computed in sequence before computing their cover query.

Table 6 (lines 2 and 4) also shows that the answer set of an $1gg$ is significantly larger when DBpedia constraints are not taken into account: the size difference goes from a small $\times 1.02$ for the homogeneous queries $Q_5$, $Q_7$ up to a striking $\times 76.42$ for the heterogeneous queries $Q_1$, $Q_4$, with a significant average of $\times 17.38$ ($\times 33.34$ for the heterogeneous queries and $\times 1.42$ for the homogeneous ones). This translates into the precision gains shown at line 5: 57.78% overall, 90.18% for the heterogeneous queries, and 25.37% for the homogeneous ones.

Conclusions: Our results confirm our claim that taking into account background knowledge yields more pregnant $1gg$s. Indeed, ontological constraints help finding common super-classes and properties to be used in $1gg$s in place of the different ones used in input queries; when constraints are ignored, these can just be generalized using variables. Therefore, the more heterogeneous input queries are, the more such common super-classes and properties are used in their $1gg$ instead of variables, and the more the gain in precision of their $1gg$ is high. For homogeneous input queries, while less striking, the gain in precision is significant in general, as our DBpedia experiments show.

7. RELATED WORK

The problem of computing an $1gg$ was introduced in the early 70’s by G. Plotkin [39, 40] to generalize First Order Logic clauses w.r.t. $\theta$-subsumption, a non-standard logical implication typical of Machine Learning. This problem has been also investigated in Knowledge Representation, for formalisms whose expressivity overlaps with our RDF setting, notably Description Logics (DLs) [29, 11, 50] and Conceptual Graphs (CGs) [17]. Finally, recently, this problem has started receiving attention in the Semantic Web field [20, 19, 31].

In DLs, computing an $1gg$ of concepts (formulæ) has been studied for $\mathcal{EL}$ and extensions thereof [29, 11, 50]. The $\mathcal{EL}$ setting translates into particular tree-shaped RDF graphs, which may feature RDFS subclass and domain constraints, and for which RDF entailment is limited to the use of these two constraints only\(^4\). In these equivalent RDF and $\mathcal{EL}$ fragments, the $\mathcal{EL}$ technique that computes an $1gg$ of $\mathcal{EL}$ concepts, which is an $\mathcal{EL}$ concept, provides only a (non least general) generalization of their corresponding tree-shaped RDF graphs w.r.t. the problem we study: the (minimal) cover graph-based $1gg$ of tree-shaped RDF graphs is clearly a forest-shaped RDF graph in general. However, it can be shown that our technique for general RDF graph can be used to compute the $\mathcal{EL}$ $1gg$ of $\mathcal{EL}$ concepts as shown in Example 1 below. Roughly speaking, given the $\mathcal{EL}$ concepts $C_1, \ldots, C_n$ and their corresponding tree-shaped RDF graphs $\mathcal{G}(C_1, b_1) , \ldots, \mathcal{G}(C_n, b_n)$, the $\mathcal{EL}$ $1gg$ is that corresponding to the tree-shaped RDF graph rooted in $b_{b_1 \ldots b_n}$ within the forest-shaped cover graph of $\mathcal{G}(C_1, b_1) , \ldots, \mathcal{G}(C_n, b_n)$.

In CGs, the so-called simple CGs with unary and binary relations correspond to particular RDF graphs (e.g., a property URI in a triple cannot be the subject or object of another triple, a class - URI or blank node - in a $\tau$ triple cannot be the subject of another $\tau$ triple nor the subject or object of another non-$\tau$ triple, etc), which may feature the four RDF constraints, and for which RDF entailment is limited to the use of these RDFS constraints only [?]. In these equivalent RDF and CG fragments, we may interchangeably compute $1gg$s with the CG technique in [17] or ours.

In RDF, computing an $1gg$ has been studied for particular RDF graphs, called $r$-graphs, ignoring RDF entailment [20, 19]. An $r$-graph is an extracted subgraph of an RDF graph $\mathcal{G}$, rooted in the $G$ value $r$ and comprising the $G$ triples reachable from $r$ through directed paths of length at most $n$. Such a rooted and directed $r$-graph can be defined recursively as $S(\mathcal{G}, r, 0)$, with:

$$S(\mathcal{G}, r, n) = \bigcup_{(v, p, v') \in G} \{ (v, p, v') \} \cup S(\mathcal{G}, r', n-1) \cup S(\mathcal{G}, p, n-1)$$

Intuitively, this purely structural definition of $r$-graph attempts carrying $G$’s knowledge about $r$. $1gg$s of $r$-graphs allow finding the commonalities between single root entities, while with general RDF graphs we further allow finding the commonalities between sets of multiple interrelated entities. The technique for computing an $1gg$ of two $r$-graphs exploits their rooted and directed structure: it starts from their respective root and traverses them simultaneously considering triples reachable through directed paths of increasing size, while incrementally constructing an $r$-graph $1gg$.

In contrast, the general RDF graphs we consider are unstructured; our technique blindly traverses the input RDF graphs to anti-unify their triples with same property, and captures their common structure across these anti-unifications thanks to the consistent naming scheme we devised for the blank nodes they generate. Further, when we ignore RDF entailment, computing $1gg$s of $r$-graphs or of RDF graphs have the same worst-case time complexity ($O(|G_1| \times |G_2|)$, with $G_1, G_2$ the input RDF graphs). This is for instance the case

\(^4\)An $\mathcal{EL}$ concept $C$ recursively translates into the RDF graph rooted in the blank node $b_r$ returned by the call $\mathcal{G}(C, b_r)$, with: $\mathcal{G}(\top, b) = \emptyset$ for the universal $\mathcal{EL}$ concept $\top$, $\mathcal{G}(A, b) = \{(\tau, \tau, A)\}$ for an atomic $\mathcal{EL}$ concept $A$, $\mathcal{G}(\exists r.C, b) = \{(b, r, b')\} \cup \mathcal{G}(C, b')$, with $b'$ a fresh blank node, for an $\mathcal{EL}$ existential restriction $\exists r.C$, and $\mathcal{G}(C_1 \cap C_2, b) = \mathcal{G}(C_1, b) \cap \mathcal{G}(C_2, b)$ for an $\mathcal{EL}$ conjunction $C_1 \cap C_2$; the $\mathcal{EL}$ constraints $A_1 \subseteq A_2$ and $\exists r.\top \subseteq A$ correspond to $(A_1, \subseteq, A_2)$ and $(r, \epsilon, A)$ resp.
8. CONCLUSION AND PERSPECTIVES

We have revisited the Machine Learning problem of computing a least general generalization (lgg) of some descriptions in the setting of RDF and SPARQL. Our contributions significantly extend the state of the art by considering the entire RDF standard and popular conjunctive fragment of SPARQL, i.e., BGPQs. In particular, we neither restrict RDF graphs and BGPQs nor RDF entailment in any way, while closely related works only consider rooted-RDF graphs and unary tree BGPQs (i.e., describing a single root resource or answer variable, respectively), and, further, they completely ignore RDF entailment rules (i.e., their particular RDF graphs and BGPQs are simply compared in a purely relational fashion). Moreover, in the case of BGPQs, we also endowed with background knowledge the standard RDF entailment relation between queries. As our experiments showed, taking into account the ontological constraints describing the application domain in which queries are posed may enhance the precision of lgg.

As a short-term perspective, we want to study heuristics in order to efficiently prune out as much as possible redundant triples, while computing lgg. Indeed, as for instance Figures 4 and 8 show, our cover graph/query technique produces many redundant triples. This would allow having more readable lgg, as well as reducing the a posteriori elimination effort of redundant triples with standard technique from the literature. Moreover, in the case of BGPQs, removing redundant triples may significantly improve their evaluation time.

Computing lgg has many possible applications in databases (recall Section 1); in particular, we plan to apply our results for BGPQs to the optimization problem of view selection in RDF. Our idea is to select as views to materialize a set $V$ of lgg of queries from a workload, such that (i) the workload queries can be (partially or totally) rewritten using $V$ and (ii) $V$ minimizes a combination of query processing, view storage, and view maintenance costs. This would provide an alternative to [24], which applies to the
so-called RDF database fragment of RDF (restricting RDF entailment) and BGPQs, and which is based on recursively decomposing the workload queries into subqueries that may be materialized.

Finally, we also want to investigate the problem of computing 1ggs, and apply it to view selection, in the setting of the DL-Lite\(_R\) description logic [15], which underpins OWL2 QL, the other W3C’s Semantic Web standard.

Acknowledgment

This work has been partially funded by Lannion-Tregor Communauté and Région Bretagne (PAWS project).

9. REFERENCES

Machine Intelligence, 5, 1970.


### APPENDIX

#### A. LUBM QUERIES

<table>
<thead>
<tr>
<th>Query</th>
<th>RDF Description</th>
</tr>
</thead>
</table>

Figure 9: LUBM queries
### B. DBPEIA QUERIES

<table>
<thead>
<tr>
<th>Query Number</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td></td>
</tr>
<tr>
<td>(?X / Y) :  -</td>
<td></td>
</tr>
<tr>
<td>Q2</td>
<td></td>
</tr>
<tr>
<td>(?X / Y) :  -</td>
<td></td>
</tr>
<tr>
<td>?X &quot;<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>&quot; &quot;<a href="http://dbpedia.org/ontology/Airport">http://dbpedia.org/ontology/Airport</a>&quot;,</td>
<td></td>
</tr>
<tr>
<td>?X &quot;<a href="http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation">http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation</a>&quot; &quot;<a href="http://dbpedia.org/resource/France">http://dbpedia.org/resource/France</a>&quot;,</td>
<td></td>
</tr>
<tr>
<td>?X &quot;<a href="http://dbpedia.org/ontology/city">http://dbpedia.org/ontology/city</a>&quot; ?Z,</td>
<td></td>
</tr>
<tr>
<td>?X &quot;<a href="http://dbpedia.org/ontology/owner">http://dbpedia.org/ontology/owner</a>&quot; ?Y,</td>
<td></td>
</tr>
<tr>
<td>?Y &quot;<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>&quot; &quot;<a href="http://dbpedia.org/ontology/Person">http://dbpedia.org/ontology/Person</a>&quot;,</td>
<td></td>
</tr>
<tr>
<td>?V &quot;<a href="http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation">http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation</a>&quot; ?X</td>
<td></td>
</tr>
<tr>
<td>Q3</td>
<td></td>
</tr>
<tr>
<td>(?X / Y) :  -</td>
<td></td>
</tr>
<tr>
<td>?X &quot;<a href="http://dbpedia.org/ontology/musicalArtist">http://dbpedia.org/ontology/musicalArtist</a>&quot; ?Y,</td>
<td></td>
</tr>
<tr>
<td>?Y &quot;<a href="http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation">http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation</a>&quot; ?Z,</td>
<td></td>
</tr>
<tr>
<td>?Y &quot;<a href="http://dbpedia.org/ontology/genre">http://dbpedia.org/ontology/genre</a>&quot; ?V,</td>
<td></td>
</tr>
<tr>
<td>Q4</td>
<td></td>
</tr>
<tr>
<td>(?X / Y) :  -</td>
<td></td>
</tr>
<tr>
<td>?X &quot;<a href="http://dbpedia.org/ontology/birthPlace">http://dbpedia.org/ontology/birthPlace</a>&quot; ?V,</td>
<td></td>
</tr>
<tr>
<td>?X &quot;<a href="http://dbpedia.org/ontology/parent">http://dbpedia.org/ontology/parent</a>&quot; ?Y,</td>
<td></td>
</tr>
<tr>
<td>?Y &quot;<a href="http://dbpedia.org/ontology/deathPlace">http://dbpedia.org/ontology/deathPlace</a>&quot; ?V,</td>
<td></td>
</tr>
<tr>
<td>?Z &quot;<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>&quot; &quot;<a href="http://dbpedia.org/ontology/Film">http://dbpedia.org/ontology/Film</a>&quot;,</td>
<td></td>
</tr>
<tr>
<td>?Z &quot;<a href="http://dbpedia.org/ontology/starring">http://dbpedia.org/ontology/starring</a>&quot; ?X,</td>
<td></td>
</tr>
<tr>
<td>Q5</td>
<td></td>
</tr>
<tr>
<td>(?X / Y) :  -</td>
<td></td>
</tr>
<tr>
<td>?X &quot;<a href="http://dbpedia.org/ontology/parent">http://dbpedia.org/ontology/parent</a>&quot; ?Y,</td>
<td></td>
</tr>
<tr>
<td>?Z &quot;<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>&quot; &quot;<a href="http://dbpedia.org/ontology/Film">http://dbpedia.org/ontology/Film</a>&quot;,</td>
<td></td>
</tr>
<tr>
<td>?Z ?V ?Y</td>
<td></td>
</tr>
<tr>
<td>Q6</td>
<td></td>
</tr>
<tr>
<td>(?X / Y) :  -</td>
<td></td>
</tr>
<tr>
<td>?X &quot;<a href="http://dbpedia.org/resource/predecessor">http://dbpedia.org/resource/predecessor</a>&quot; ?Y,</td>
<td></td>
</tr>
<tr>
<td>?Z &quot;<a href="http://dbpedia.org/ontology/creator">http://dbpedia.org/ontology/creator</a>&quot; ?X,</td>
<td></td>
</tr>
<tr>
<td>Q7</td>
<td></td>
</tr>
<tr>
<td>(?X / Y) :  -</td>
<td></td>
</tr>
<tr>
<td>?X &quot;<a href="http://dbpedia.org/resource/parent">http://dbpedia.org/resource/parent</a>&quot; ?Y,</td>
<td></td>
</tr>
<tr>
<td>?Z &quot;<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>&quot; &quot;<a href="http://dbpedia.org/ontology/Film">http://dbpedia.org/ontology/Film</a>&quot;,</td>
<td></td>
</tr>
<tr>
<td>?Z &quot;<a href="http://dbpedia.org/ontology/starring">http://dbpedia.org/ontology/starring</a>&quot; ?X,</td>
<td></td>
</tr>
<tr>
<td>Q8</td>
<td></td>
</tr>
<tr>
<td>(?X / Y) :  -</td>
<td></td>
</tr>
<tr>
<td>?X &quot;<a href="http://dbpedia.org/resource/parent">http://dbpedia.org/resource/parent</a>&quot; ?Y,</td>
<td></td>
</tr>
<tr>
<td>?Z &quot;<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>&quot; &quot;<a href="http://dbpedia.org/ontology/Film">http://dbpedia.org/ontology/Film</a>&quot;,</td>
<td></td>
</tr>
<tr>
<td>?Z &quot;<a href="http://dbpedia.org/ontology/starring">http://dbpedia.org/ontology/starring</a>&quot; ?X,</td>
<td></td>
</tr>
</tbody>
</table>

Figure 10: DBPEDIA queries