Least general generalizations in RDF and SPARQL

Sara El Hassad, François Goasdoué, Hélène Jaudoin

Least general generalizations in RDF and SPARQL

Technical report associated to a submission, made in early september 2016, to an international venue

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 commonalities between descriptions of data and knowledge is an old problem in Machine Learning. This was formalized by G. Plotkin in the early 70's, as computing *least general generalizations* (`lgg`s) of First Order Logic formulae [21, 22].

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. Our contributions to this problem are:

1. Our technique for computing `lgg`s of RDF graphs is the first that takes *entirely* into account the RDF standard, i.e., we do not restrict RDF graphs or RDF reasoning in any way, while the state of the art imposes significant limitations [10]: RDF graphs are almost-trees and RDF reasoning is ignored.

2. Our technique for computing `lgg`s of SPARQL queries is also the first that takes *entirely* into account the well-established conjunctive fragment of the SPARQL standard, a.k.a. Basic Graph Pattern Queries (BGPQs), while the related work in the literature only considers unary tree shaped BGPQs [15]. Further, when available, we devise how to incorporate background knowledge, formalized as ontological constraints modeling the application domain, in order to compute much more pregnant `lgg`s.

3. We provide an experimental assessment of the above results. We shows, in particular, that `lgg`s of BGPQs endowed with background knowledge are much more precise than those computed while ignoring such extra knowledge.

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. We report on the experiments we conducted in Section 5. Finally, we present related works in Section 6 and we conclude and draw some perspectives in Section 7.

RDF statement	Triple	Relational assertion
Class assertion	$(\mathtt{s}, \text{rdf:type}, \mathtt{o})$	$\mathtt{o}(\mathtt{s})$
Property assertion	$(\mathtt{s}, \mathtt{p}, \mathtt{o})$ with $\mathtt{p} \neq \text{rdf:type}$	$\mathtt{p}(\mathtt{s}, \mathtt{o})$

RDFS statement	Triple	Relational constraint
Subclass	$(\mathtt{s}, \text{rdfs:subClassOf}, \mathtt{o})$	$\mathtt{s} \subseteq \mathtt{o}$
Subproperty	$(\mathtt{s}, \text{rdfs:subPropertyOf}, \mathtt{o})$	$\mathtt{s} \subseteq \mathtt{o}$
Domain typing	$(\mathtt{s}, \text{rdfs:domain}, \mathtt{o})$	$\Pi_{\text{domain}}(\mathtt{s}) \subseteq \mathtt{o}$
Range typing	$(\mathtt{s}, \text{rdfs:range}, \mathtt{o})$	$\Pi_{\text{range}}(\mathtt{s}) \subseteq \mathtt{o}$

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

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 $(\mathtt{s}, \mathtt{p}, \mathtt{o})$. A triple states that its *subject* $\mathtt{s}$ has the *property* $\mathtt{p}$, the value of which is the *object* $\mathtt{o}$. Triples are built using three pairwise disjoint sets: a set $\mathcal{U}$ of *uniform resources identifiers (URIs)*, a set $\mathcal{L}$ of *literals* (constants), and a set $\mathcal{B}$ of *blank nodes* allowing to support *incomplete information*. Blank nodes can be seen as identifiers for missing values (unknown URIs or literals); they are the RDF counterparts of labelled nulls or existential variables in incomplete relational databases [14, 1], as shown in [12]. *Well-formed triples*, as per the RDF specification [25], belong to $(\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{L} \cup \mathcal{B})$; we only consider such triples hereafter.

Notations. We use $\mathtt{s}$, $\mathtt{p}$, $\mathtt{o}$ in triples as placeholders. We note $\mathtt{Val}(\mathcal{G})$ the set of *values* occurring in an RDF graph $\mathcal{G}$, i.e., the URIs, literals and blank nodes; we note $\mathtt{Bl}(\mathcal{G})$ 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 $(b, \text{hasTitle}, \text{“LGG in RDF”})$ and $(b, \text{hasContactAuthor}, b_1)$ state that *something (b) entitled “LGG in RDF” has somebody (b_1) as contact author*.

A triple models 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 [25] 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 $(b, \text{rdf:type}, \text{ConfPaper})$.

2.1.2 Adding ontological knowledge to RDF graphs

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 buit-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 $(\text{ConfPaper}, \text{rdfs:subClassOf}, \text{Publication})$ states that *conference papers are publications*, the triple $(\text{hasContactAuthor}, \text{rdfs:subPropertyOf}, \text{hasAuthor})$ states that *having a contact author is having an author*, the triple $(\text{hasAuthor}, \text{rdfs:domain}, \text{Publication})$ states that *only publications have authors* while the triple $(\text{hasAuthor}, \text{rdfs:range}, \text{Researcher})$ states that *only researchers are authors of something*.

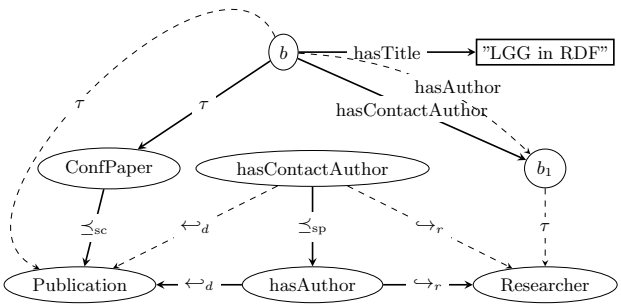

Figure 1: Sample RDF graph $\mathcal{G}$.

Notations. From now, for conciseness, we use the following shorthands for built-in properties: τ for rdf:type, $\prec_{\text{sc}}$ for rdfs:subClassOf, $\prec_{\text{sp}}$ for rdfs:subPropertyOf, $\hookleftarrow_d$ for rdfs:domain, and $\hookrightarrow_r$ 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 $(\mathtt{s}, \mathtt{p}, \mathtt{o})$ corresponds to the $\mathtt{p}$-labeled directed edge from the $\mathtt{s}$ node to the $\mathtt{o}$ 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 $(\text{hasContactAuthor}, \text{rdfs:subPropertyOf}, \text{hasAuthor})$ is violated under CWA, as the triple $(b, \text{hasContactAuthor}, b_1)$ is explicit in $\mathcal{G}$ while $(b, \text{hasAuthor}, b_1)$ is not. By contrast, under OWA, the same constraint allows deriving the “missing” triple $(b, \text{hasAuthor}, b_1)$, making it implicit within $\mathcal{G}$. Further, this implicit triple together with the explicit ontological constraint $(\text{hasAuthor}, \text{rdfs:range}, \text{Researcher})$ yields the implicit 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., *RDFS statements*). They encode the *propagation* of assertions through constraints (`rdfs2`, `rdfs3`, `rdfs7`, `rdfs9`), the *transitivity* of the $\prec_{\text{sp}}$ and $\prec_{\text{sc}}$ constraints (`rdfs5`, `rdfs11`), the *complementation* of domains or ranges through $\prec_{\text{sc}}$ (`ext1`, `ext2`), and the *inheritance* of domains and of ranges through $\prec_{\text{sp}}$ (`ext3`, `ext4`).

Rule name [26]	Entailment rule
rdfs2	$(\mathtt{p}, \hookrightarrow_d, \mathtt{o}), (\mathtt{s}_1, \mathtt{p}, \mathtt{o}_1) \rightarrow (\mathtt{s}_1, \tau, \mathtt{o})$
rdfs3	$(\mathtt{p}, \hookrightarrow_r, \mathtt{o}), (\mathtt{s}_1, \mathtt{p}, \mathtt{o}_1) \rightarrow (\mathtt{o}_1, \tau, \mathtt{o})$
rdfs5	$(\mathtt{p}_1, \preceq_{\text{sp}}, \mathtt{p}_2), (\mathtt{p}_2, \preceq_{\text{sp}}, \mathtt{p}_3) \rightarrow (\mathtt{p}_1, \preceq_{\text{sp}}, \mathtt{p}_3)$
rdfs7	$(\mathtt{p}_1, \preceq_{\text{sp}}, \mathtt{p}_2), (\mathtt{s}, \mathtt{p}_1, \mathtt{o}) \rightarrow (\mathtt{s}, \mathtt{p}_2, \mathtt{o})$
rdfs9	$(\mathtt{s}, \preceq_{\text{sc}}, \mathtt{o}), (\mathtt{s}_1, \tau, \mathtt{s}) \rightarrow (\mathtt{s}_1, \tau, \mathtt{o})$
rdfs11	$(\mathtt{s}, \preceq_{\text{sc}}, \mathtt{o}), (\mathtt{o}, \preceq_{\text{sc}}, \mathtt{o}_1) \rightarrow (\mathtt{s}, \preceq_{\text{sc}}, \mathtt{o}_1)$
ext1	$(\mathtt{p}, \hookrightarrow_d, \mathtt{o}), (\mathtt{o}, \preceq_{\text{sc}}, \mathtt{o}_1) \rightarrow (\mathtt{p}, \hookrightarrow_d, \mathtt{o}_1)$
ext2	$(\mathtt{p}, \hookrightarrow_r, \mathtt{o}), (\mathtt{o}, \preceq_{\text{sc}}, \mathtt{o}_1) \rightarrow (\mathtt{p}, \hookrightarrow_r, \mathtt{o}_1)$
ext3	$(\mathtt{p}, \preceq_{\text{sp}}, \mathtt{p}_1), (\mathtt{p}_1, \hookrightarrow_d, \mathtt{o}) \rightarrow (\mathtt{p}, \hookrightarrow_d, \mathtt{o})$
ext4	$(\mathtt{p}, \preceq_{\text{sp}}, \mathtt{p}_1), (\mathtt{p}_1, \hookrightarrow_r, \mathtt{o}) \rightarrow (\mathtt{p}, \hookrightarrow_r, \mathtt{o})$

Table 2: Sample set of RDF entailment rules.

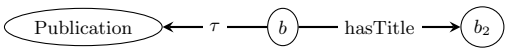

Figure 2: Sample RDF graph $\mathcal{G}'$.

The *saturation (a.k.a. closure)* of an RDF graph $\mathcal{G}$ w.r.t. a set $\mathcal{R}$ of RDF entailment rules, is the RDF graph $\mathcal{G}^\infty$ obtained by adding to $\mathcal{G}$ all the implicit triples that follow from $\mathcal{G}$ and $\mathcal{R}$. Roughly speaking, the saturation $\mathcal{G}^\infty$ *materializes* the semantics of $\mathcal{G}$. It corresponds to the fixed-point obtained by repeatedly applying the rules in $\mathcal{R}$ to $\mathcal{G}$ in a forward-chaining fashion, a.k.a. *chasing* $\mathcal{G}$ with $\mathcal{R}$ in the database parlance. In RDF, the saturation is *always* finite and unique (up to blank node renaming), and does not hold implicit triples.

The saturation of the RDF graph $\mathcal{G}$ shown in Figure 1 corresponds to the RDF graph $\mathcal{G}^\infty$ in which all the $\mathcal{G}$ implicit triples have been made explicit. It is worth noting how, starting from $\mathcal{G}$, applying RDF entailment rules *mechanizes* the construction of $\mathcal{G}^\infty$. 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 $(b, \text{hasContactAuthor}, b_1)$ trigger the `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 `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 $\mathcal{G}$ is more specific than another RDF graph $\mathcal{G}'$, or equivalently $\mathcal{G}'$ is more general than $\mathcal{G}$, whenever there is an embedding of $\mathcal{G}'$ into the *saturation* of $\mathcal{G}$, i.e., the complete set of triples that $\mathcal{G}$ models.

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

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

Notations. When relevant to the discussion, we designate by $\mathcal{G} \models^\phi_\mathcal{R} \mathcal{G}'$ the fact that the entailment $\mathcal{G} \models_\mathcal{R} \mathcal{G}'$ holds due to the graph homomorphism ϕ. Also, when RDF entailment rules are disregarded, i.e., $\mathcal{R} = \emptyset$, 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 $\mathtt{Triple}(\mathtt{s}, \mathtt{p}, \mathtt{o})$ relation.

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

1. $\mathcal{G}$ and $\mathcal{G}^\infty$ are equivalent, noted $\mathcal{G} \equiv_\mathcal{R} \mathcal{G}^\infty$, since clearly $\mathcal{G} \models_\mathcal{R} \mathcal{G}^\infty$ and $\mathcal{G}^\infty \models_\mathcal{R} \mathcal{G}$ holds,
2. $\mathcal{G} \models_\mathcal{R} \mathcal{G}'$ iff $\mathcal{G}^\infty \models \mathcal{G}'$ 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 (BGPQs)*, is the counterpart of the relational select-project-join queries; it is the most widely used subset of SPARQL queries in real-world applications [18].

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 $\mathcal{V}$ of variables, pairwise disjoint with $\mathcal{U}$, $\mathcal{L}$ and $\mathcal{B}$, triple patterns belong to: $(\mathcal{V} \cup \mathcal{U} \cup \mathcal{B}) \times (\mathcal{V} \cup \mathcal{U}) \times (\mathcal{V} \cup \mathcal{U} \cup \mathcal{L} \cup \mathcal{B})$.

Notations. We adopt the usual conjunctive query notation $q(\bar{x}) \leftarrow t_1, \ldots, t_\alpha$, where $\{t_1, \ldots, t_\alpha\}$ is a BGP; the query head variables $\bar{x}$ are called *answer variables*, and form a subset of the variables occurring in $t_1, \ldots, t_\alpha$; for boolean queries, $\bar{x}$ is empty. The head of q, noted $head(q)$, is $q(\bar{x})$, and the body of q, denoted $body(q)$, is the BGP $\{t_1, \ldots, t_\alpha\}$. We use x and y in queries, possibly with subscripts, for answer and non-answer variables respectively. Finally, we denote by $\mathtt{VarBl}(q)$ the set of variables *and* blank nodes occurring in the query q; we note $\mathtt{Val}(q)$ 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 $\mathcal{G}$ and a set $\mathcal{R}$ of RDF entailment rules, $\mathcal{G}$ *entails* q, denoted $\mathcal{G} \models_\mathcal{R} q$, iff $\mathcal{G} \models_\mathcal{R} body(q)$ holds, i.e., there exists a homomorphism ϕ from $\mathtt{VarBl}(q)$ to $\mathtt{Val}(\mathcal{G}^\infty)$ such that $[body(q)]_\phi \subseteq \mathcal{G}^\infty$.

The RDF graph $\mathcal{G}$ 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 ϕ 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 ϕ already holds for $\mathcal{R} = \emptyset$, i.e., considering only the explicit triples in Figure 1.

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

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 $\mathcal{G}$* is:

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

where we denote by $(\bar{x})_{\phi}$ the tuple of $\mathcal{G}^{\infty}$ 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 $\mathcal{G}$ 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.

Finally, remark that query entailment and query answering *treat the blank nodes in a query exactly as non-answer variables* [27].

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 $body(q) \models^{\phi}_{\mathcal{R}} body(q')$ *with* $(\bar{x}')_{\phi} = \bar{x}$ holds. Here, $body(q) \models^{\phi}_{\mathcal{R}} body(q')$ is the obvious adaptation of the above-mentioned entailment relationships between RDF graphs to the fact that the query bodies may feature variables, i.e., ϕ is a homomorphism from $\texttt{VarBl}(q')$ to $\texttt{Val}(q)$ such that $[body(q')]_{\phi} \subseteq body(q)^{\infty}$.

For instance, the query $q_1(x)$:- $(x, \tau, \text{ConfPaper})$, $(x, \text{hasContactAuthor}, y)$ entails the query $q_2(x)$:- (x, τ, y) with $\phi(x) = x$, $\phi(y) = \text{ConfPaper}$ and any entailment rule set. However, it is worth noting that, *counterintuitively*, 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).

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* (`lgg` for short). Then, we devise a technique for computing such a `lgg` in Section 3.2.

3.1 Defining the `lgg` of RDF graphs

A *least general generalization* of n descriptions $d_1, \ldots, d_n$ is a most specific description d generalizing every $d_{1 \leq i \leq n}$ for some generalization/specialization relation between descriptions [21, 22]. In our RDF setting, we use RDF graphs as descriptions and entailment between RDF graphs as relation for generalization/specialization:

DEFINITION 1 (`lgg` OF RDF GRAPHS). *Let $\mathcal{G}_1, \ldots, \mathcal{G}_n$ be RDF graphs and $\mathcal{R}$ a set of RDF entailment rules.*

- *A* generalization *of $\mathcal{G}_1, \ldots, \mathcal{G}_n$ is an RDF graph $\mathcal{G}_g$ such that $\mathcal{G}_i \models_{\mathcal{R}} \mathcal{G}_g$ holds for $1 \leq i \leq n$.*
- *A* least general generalization *(`lgg`) of $\mathcal{G}_1, \ldots, \mathcal{G}_n$ is a generalization $\mathcal{G}_{\texttt{lgg}}$ of $\mathcal{G}_1, \ldots, \mathcal{G}_n$ such that for any other generalization $\mathcal{G}_g$ of $\mathcal{G}_1, \ldots, \mathcal{G}_n$, $\mathcal{G}_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}_g$ holds.*

Observe that the above definition indicates that an `lgg` of RDF graphs is *unique* up to entailment (since $\mathcal{G}_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}_g$ holds for any $\mathcal{G}_g$). Indeed, if it were that RDF graphs have *multiple* `lgg`s *incomparable* w.r.t. entailment, say $\texttt{lgg}_1$, ..., $\texttt{lgg}_m$, their merge[1] $\texttt{lgg}_1 \uplus \cdots \uplus \texttt{lgg}_m$ would be a *single strictly more specific* `lgg`, a contradiction.

Figure 3 displays two RDF graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, as well as their minimal `lgg` (with lowest number of triples) when we consider the RDF entailment rules shown in Table 2: $\mathcal{G}_{\texttt{lgg}}$. $\mathcal{G}_1$ describes a conference paper i_1 with title "Disaggregations in Databases" and author Serge Abiteboul, who is a researcher; also conference papers are publications. $\mathcal{G}_2$ describes a journal paper i_2 with title "Computing with First-Order Logic", contact author Serge Abiteboul and author Victor Vianu, who are researchers; moreover, journal papers are publications and having a contact author is having an author. $\mathcal{G}_{\texttt{lgg}}$ states that their common information comprises the existence of a resource $(b_{i_1 i_2})$ having some type $(b_{\text{C(onf)P(aper)J(our)P(aper)}})$, which is a particular case of publication, with some title $(b_{\text{D(iD)C(wFOL)}})$ 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 under consideration allows deriving them from the remaining triples (e.g., materializing the only $\mathcal{G}_{\texttt{lgg}}$ implicit triple in Figure 3 would make it redundant if we consider the RDF entailment rules in Table 2) or triples generalizing others without needing RDF entailment rules, i.e., in a purely relational fashion (e.g., adding the triple $(b, \text{hasAuthor}, b')$ to $\mathcal{G}_{\texttt{lgg}}$ in Figure 3 would be redundant w.r.t. $(b_{i_1 i_2}, \text{hasAuthor}, SA)$). Also, an `lgg` may have *several minimal* syntactical variants obtained by pruning out redundant triples. For example, think of a minimal `lgg` comprising the triples $(A, \preceq_{\text{sc}}, B)$, $(B, \preceq_{\text{sc}}, A)$ and (b, τ, A), i.e., there exists an instance of the

[1]The merge of RDF graphs, performed with the RDF specific merge operator $\uplus$, 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).

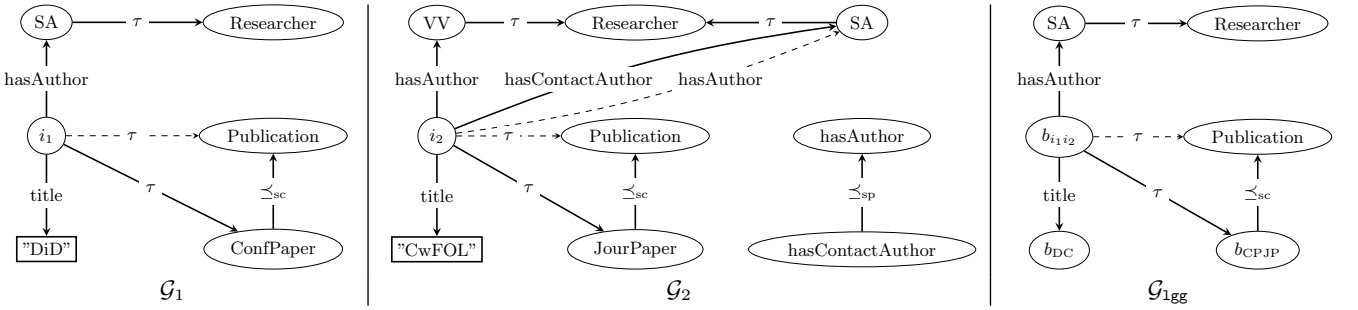

Figure 3: Sample RDF graphs $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_{\texttt{lgg}}$, with $\mathcal{G}_{\texttt{lgg}}$ the minimal `lgg` of $\mathcal{G}_1$ and $\mathcal{G}_2$.

class A, which is equivalent to class B. Clearly, an equivalent and minimal variant of this `lgg` is the RDF graph comprising the triples $(A, \preceq_{\text{sc}}, B)$, $(B, \preceq_{\text{sc}}, A)$ and (b, τ, B).

Importantly, the above discussion is not specific to `lgg`s of RDF graphs, since any RDF graph may feature redundancy. The detection and elimination of RDF graph redundancy has been studied in the literature, e.g., [16, 19, 20], hence we focus in this work on finding *some* `lgg` of RDF graphs.

The proposition below shows that an `lgg` of n RDF graphs, with $n \geq 3$, can be defined (hence computed) as a sequence of $n-1$ `lgg`s of *two* RDF graphs. Intuitively, assuming that $\ell_{k\geq 2}$ is an operator computing an `lgg` of k input RDF graphs, the next proposition establishes that:

$$\begin{array}{ll} \ell_3(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3) & \equiv_{\mathcal{R}} \ \ell_2(\ell_2(\mathcal{G}_1, \mathcal{G}_2), \mathcal{G}_3) \\ \dots & \dots \\ \ell_n(\mathcal{G}_1, \dots, \mathcal{G}_n) & \equiv_{\mathcal{R}} \ \ell_2(\ell_{n-1}(\mathcal{G}_1, \dots, \mathcal{G}_{n-1}), \mathcal{G}_n) \\ & \equiv_{\mathcal{R}} \ \ell_2(\ell_2(\cdots \ell_2(\ell_2(\mathcal{G}_1, \mathcal{G}_2), \mathcal{G}_3) \cdots, \mathcal{G}_{n-1}), \mathcal{G}_n) \end{array}$$

Proposition 1. *Let $\mathcal{G}_1, \dots, \mathcal{G}_{n\geq 3}$ be n RDF graphs and $\mathcal{R}$ a set of RDF entailment rules. $\mathcal{G}_{\texttt{lgg}}$ is an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_n$ iff $\mathcal{G}_{\texttt{lgg}}$ is an `lgg` of an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_{n-1}$ and $\mathcal{G}_n$.*

Proof. The proof relies on the next lemma.

Lemma 1. *Let $\mathcal{G}_1, \dots, \mathcal{G}_n$ be RDF graphs and $\mathcal{R}$ a set of RDF entailment rules. If $\mathcal{G}^1_{\texttt{lgg}}$ is an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_{k<n}$ and $\mathcal{G}^2_{\texttt{lgg}}$ is an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_n$, then $\mathcal{G}^1_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}^2_{\texttt{lgg}}$ holds.*

Let us show that the above lemma holds.

Suppose that $\mathcal{G}^1_{\texttt{lgg}}$ is an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_{k<n}$, i.e., by Definition 1: (i) $\mathcal{G}_{1\leq i\leq k} \models_{\mathcal{R}} \mathcal{G}^1_{\texttt{lgg}}$ holds and (ii) for any RDF graph $\mathcal{G}$ such that $\mathcal{G}_{1\leq i\leq k} \models_{\mathcal{R}} \mathcal{G}$ holds, $\mathcal{G}^1_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}$ holds.

Suppose also that $\mathcal{G}^2_{\texttt{lgg}}$ is an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_n$, i.e., by Definition 1: (i) $\mathcal{G}_{1\leq i\leq n} \models_{\mathcal{R}} \mathcal{G}^2_{\texttt{lgg}}$ holds and (ii) for any RDF graph $\mathcal{G}'$ such that $\mathcal{G}_{1\leq i\leq n} \models_{\mathcal{R}} \mathcal{G}'$ holds, $\mathcal{G}^2_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}'$ holds.

Clearly, $\mathcal{G}^2_{\texttt{lgg}}$ is a possible value for $\mathcal{G}$ above, because $\mathcal{G}_{1\leq i\leq n} \models_{\mathcal{R}} \mathcal{G}^2_{\texttt{lgg}}$ implies $\mathcal{G}_{1\leq i\leq k<n} \models_{\mathcal{R}} \mathcal{G}^2_{\texttt{lgg}}$, hence $\mathcal{G}^1_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}^2_{\texttt{lgg}}$ must hold. $\lhd$

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

($\Rightarrow$) Assume that $\mathcal{G}_{\texttt{lgg}}$ is an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_n$ and let us show that it is also an `lgg` of some `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_{n-1}$ and $\mathcal{G}_n$. By Definition 1, because $\mathcal{G}_{\texttt{lgg}}$ is an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_n$, we have: (i) $\mathcal{G}_{1\leq i\leq n} \models_{\mathcal{R}} \mathcal{G}_{\texttt{lgg}}$ holds and (ii) for any RDF graph $\mathcal{G}$ such that $\mathcal{G}_{1\leq i\leq n} \models_{\mathcal{R}} \mathcal{G}$ holds, $\mathcal{G}_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}$ holds.

Let $\mathcal{G}'_{\texttt{lgg}}$ be some `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_{n-1}$. From (i) and the above lemma, (*) $\mathcal{G}'_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}_{\texttt{lgg}}$ and $\mathcal{G}_n \models_{\mathcal{R}} \mathcal{G}_{\texttt{lgg}}$ holds.

Moreover, for any RDF graph $\mathcal{G}$, if it were that $\mathcal{G}'_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}$ and $\mathcal{G}_n \models_{\mathcal{R}} \mathcal{G}$ and $\mathcal{G}_{\texttt{lgg}} \not\models_{\mathcal{R}} \mathcal{G}$, then $\mathcal{G}_{1\leq i\leq n} \models_{\mathcal{R}} \mathcal{G}$ would hold and contradict the fact that $\mathcal{G}_{\texttt{lgg}}$ is an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_n$. Therefore, (**) for any RDF graph $\mathcal{G}$, if $\mathcal{G}'_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}$ and $\mathcal{G}_n \models_{\mathcal{R}} \mathcal{G}$ holds, then $\mathcal{G}_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}$ holds.

From Definition 1, (*) and (**) we get that $\mathcal{G}_{\texttt{lgg}}$ is an `lgg` of an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_{n-1}$ and $\mathcal{G}_n$.

($\Leftarrow$) Assume that $\mathcal{G}_{\texttt{lgg}}$ is an `lgg` of an `lgg` $\mathcal{G}'$ of $\mathcal{G}_1, \dots, \mathcal{G}_{n-1}$ and $\mathcal{G}_n$, and let us show that it is also an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_n$.

By Definition 1, (i) $\mathcal{G}' \models_{\mathcal{R}} \mathcal{G}_{\texttt{lgg}}$ and $\mathcal{G}_n \models_{\mathcal{R}} \mathcal{G}_{\texttt{lgg}}$ hold, hence $\mathcal{G}_{1\leq i\leq n-1} \models_{\mathcal{R}} \mathcal{G}_{\texttt{lgg}}$ and $\mathcal{G}_n \models_{\mathcal{R}} \mathcal{G}_{\texttt{lgg}}$ hold, i.e., (*') $\mathcal{G}_{1\leq i\leq n} \models_{\mathcal{R}} \mathcal{G}_{\texttt{lgg}}$ holds.

Moreover, (ii) for any RDF graph $\mathcal{G}$ such that $\mathcal{G}' \models_{\mathcal{R}} \mathcal{G}$ and $\mathcal{G}_n \models_{\mathcal{R}} \mathcal{G}$ hold, $\mathcal{G}_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}$ holds. By Definition 1, $\mathcal{G}_{1\leq i\leq n-1} \models_{\mathcal{R}} \mathcal{G}'$ holds, therefore we get: (**') for any RDF graph $\mathcal{G}$ such that $\mathcal{G}_{1\leq i\leq n} \models_{\mathcal{R}} \mathcal{G}$ holds, $\mathcal{G}_{\texttt{lgg}} \models_{\mathcal{R}} \mathcal{G}$ holds.

From Definition 1, (*') and (**') we get that $\mathcal{G}_{\texttt{lgg}}$ is an `lgg` of $\mathcal{G}_1, \dots, \mathcal{G}_n$. $\square$

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 $\mathcal{G}_1, \mathcal{G}_2$ and a set $\mathcal{R}$ of RDF entailment rules, we want to compute* some `lgg` *of $\mathcal{G}_1$ and $\mathcal{G}_2$.*

3.2 Computing an `lgg` of RDF graphs

We first devise the *cover graph* of two RDF graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ (to be defined shortly, Definition 2 below), which is central to our technique for computing an `lgg` of $\mathcal{G}_1$ and $\mathcal{G}_2$. We indeed show (Theorem 1) that this particular RDF graph corresponds to an `lgg` of $\mathcal{G}_1$ and $\mathcal{G}_2$ when considering their explicit triples *only*, i.e., ignoring RDF entailment rules. Then, we show the main result of this section (Theorem 2): an `lgg` of $\mathcal{G}_1$ and $\mathcal{G}_2$, for *any set* $\mathcal{R}$ of RDF entailment rules, is the cover graph of their saturations w.r.t. $\mathcal{R}$. We also provide the worst-case size of cover graph-based `lgg`s, as well as the worst-case time to compute them.

Definition 2 (Cover graph). *The* cover graph $\mathcal{G}$ *of two RDF graph $\mathcal{G}_1$ and $\mathcal{G}_2$ is the RDF graph such that for every property p in both $\mathcal{G}_1$ and $\mathcal{G}_2$:*

$$(t_1, p, t_2) \in \mathcal{G}_1 \text{ and } (t_3, p, t_4) \in \mathcal{G}_2 \text{ iff } (t_5, p, t_6) \in \mathcal{G}$$

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

The cover graph is more general than $\mathcal{G}_1$ and $\mathcal{G}_2$ as each of its triple (t_5, p, t_6) is a *least general anti-unifier* of a triple (t_1, p, t_2) from $\mathcal{G}_1$ and a triple (t_3, p, t_4) from $\mathcal{G}_2$. The notion

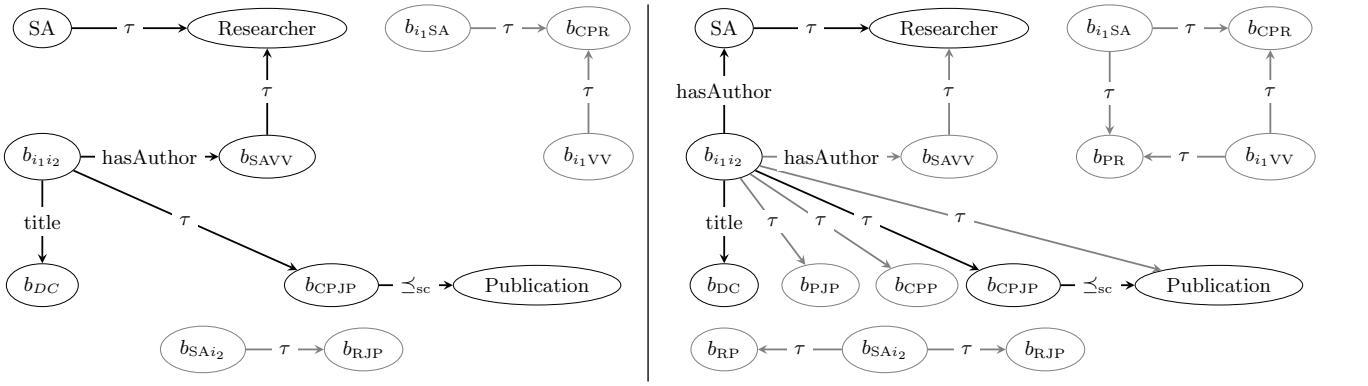

Figure 4: Cover graphs of $\mathcal{G}_1$ and $\mathcal{G}_2$ in Figure 3 (left) and of their saturations w.r.t. the entailment rules in Table 2 (right). Triples shown in grey are redundant w.r.t. those shown in black.

of *least general anti-unifier* [21, 22, 24] is dual to the well-known notion of *most general unifier* [23, 24]. Observe that $\mathcal{G}$'s triples result from anti-unifications of $\mathcal{G}_1$ and $\mathcal{G}_2$ triples with *same* property URI. Indeed, anti-unifying triples of the form $(\mathtt{s}_1, p, \mathtt{o}_1)$ and $(\mathtt{s}_2, p', \mathtt{o}_2)$, with $p \neq p'$, would lead to a non-well-formed triples of the form $(\mathtt{s}, b_{pp'}, \mathtt{o})$ (recall that property values *must* be URIs in RDF graphs), where $b_{pp'}$ is the blank node required to generalize the distinct values p and p'.

Further, the cover graph is an `lgg` for the explicit triples in $\mathcal{G}_1$ and those in $\mathcal{G}_2$ since, intuitively, we capture their *common structures* by consistently naming, across all the anti-unifications begetting $\mathcal{G}$, the blank nodes used to generalize pairs of distinct subject values or of object values: each time the distinct values t from $\mathcal{G}_1$ and t' from $\mathcal{G}_2$ are generalized by a blank node while anti-unifying two triples, it is *always* by the same blank node $b_{t,t'}$ in $\mathcal{G}$. This way, we establish *joins* between $\mathcal{G}$ triples, which reflect the common join structure on t within $\mathcal{G}_1$ and on t' within $\mathcal{G}_2$. For instance in Figure 3, the *explicit* triples $(i_1, \tau, \text{ConfPaper})$, $(\text{ConfPaper}, \preceq_{\text{sc}}, \text{Publication})$, $(i_1, \text{title}, \text{"DiD"})$ in $\mathcal{G}_1$, and $(i_2, \tau, \text{JourPaper})$, $(\text{JourPaper}, \preceq_{\text{sc}}, \text{Publication})$, $(i_2, \text{title}, \text{"CwFOL"})$ in $\mathcal{G}_2$, lead to the triples $(b_{i_1i_2}, \tau, b_{\text{CPJP}})$, $(b_{\text{CPJP}}, \preceq_{\text{sc}}, \text{Publication})$, $(b_{i_1i_2}, \text{title}, b_{\text{DC}})$ in the cover graph of $\mathcal{G}_1$ and $\mathcal{G}_2$ shown in Figure 4 (left). The first above-mentioned $\mathcal{G}$ triple results from anti-unifying i_1 and i_2 into $b_{i_1i_2}$, and, ConfPaper and JourPaper into b_{CPJP}. The second results from anti-unifying *again* ConfPaper and JourPaper into b_{CPJP}, and, Publication and Publication into Publication (as a constant is its own least general generalization). Finally, the third results from anti-unifying *again* i_1 and i_2 into $b_{i_1i_2}$, and, "DiD" and "CwFOL" into b_{DC}. By reusing consistently the same blank node name $b_{i_1i_2}$ for each anti-unification of the constants i_1 and i_2 (resp. b_{CPJP} for ConfPaper and JourPaper)), the cover graph triples join on $b_{i_1i_2}$ (resp. b_{CPJP}) in order to reflect that, in $\mathcal{G}_1$ and in $\mathcal{G}_2$, there exists a particular case of publication (i_1 in $\mathcal{G}_1$ and i_2 in $\mathcal{G}_2$) with some title ("DiD" in $\mathcal{G}_1$ and "CwFOL" in $\mathcal{G}_2$).

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

THEOREM 1. *The* cover graph $\mathcal{G}$ *of the RDF graphs* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *is an* `lgg` *of them for the empty set* $\mathcal{R}$ *of RDF entailment rules (i.e.,* $\mathcal{R} = \emptyset$*).*

PROOF. The proof can be directly derived from that of the more general Theorem 2 (see below), noting that in the particular case of Theorem 1 $\mathcal{R} = \emptyset$ holds, hence $\mathcal{G}_1^\infty = \mathcal{G}_1$ and $\mathcal{G}_2^\infty = \mathcal{G}_2$ holds. □

We provide below worst-case bounds for the time to compute a cover graph and for its size. Clearly, these bounds are met when all the triples of the two input graphs use the same property URI.

PROPOSITION 2. *The cover graph of two RDF graphs* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *can be computed in* $O(|\mathcal{G}_1| \times |\mathcal{G}_2|)$*; its size is bounded by* $|\mathcal{G}_1| \times |\mathcal{G}_2|$.

The main theorem below generalizes Theorem 1 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 2. *Let* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *be two RDF graphs, and* $\mathcal{R}$ *a set of RDF entailment rules. The* cover graph $\mathcal{G}$ *of* $\mathcal{G}_1^\infty$ *and* $\mathcal{G}_2^\infty$ *is an* `lgg` *of* $\mathcal{G}_1$ *and* $\mathcal{G}_2$.

PROOF. We start by showing that $\mathcal{G}$ is a generalization of $\mathcal{G}_1$ and $\mathcal{G}_2$, i.e., $\mathcal{G}_1 \models_{\mathcal{R}} \mathcal{G}$ *and* that $\mathcal{G}_2 \models_{\mathcal{R}} \mathcal{G}$. Consider the RDF graph $\mathcal{G}'$ obtained from $\mathcal{G}$ by replacing every blank node $b_{v_1v_2}$ by the value v_1. Clearly, $\mathcal{G}' \models_{\mathcal{R}} \mathcal{G}$ and, by construction of $\mathcal{G}$, $\mathcal{G}' = \mathcal{G}_1$, i.e., $\mathcal{G}_1 \models_{\mathcal{R}} \mathcal{G}$. Showing $\mathcal{G}_2 \models_{\mathcal{R}} \mathcal{G}$ is done in a similar way. Hence, $\mathcal{G}$ is a generalization of $\mathcal{G}_1$ and $\mathcal{G}_2$.

Let us show now that $\mathcal{G}$ is an `lgg` of $\mathcal{G}_1$ and $\mathcal{G}_2$. Let $\mathcal{G}_{\mathtt{lgg}}$ be *any* `lgg` of $\mathcal{G}_1$ and $\mathcal{G}_2$. To prove our claim, we need to show that $\mathcal{G} \models_{\mathcal{R}} \mathcal{G}_{\mathtt{lgg}}$ holds. Since $\mathcal{G}_{\mathtt{lgg}}$ is an `lgg` of $\mathcal{G}_1$ and $\mathcal{G}_2$, there exist two homomorphisms ϕ_1 and ϕ_2 from the blank nodes in $\mathcal{G}_{\mathtt{lgg}}$ to the values in $\mathcal{G}_1^\infty$ and in $\mathcal{G}_2^\infty$ respectively, such that $[\mathcal{G}_{\mathtt{lgg}}]_{\phi_1} \subseteq \mathcal{G}_1^\infty$ and $[\mathcal{G}_{\mathtt{lgg}}]_{\phi_2} \subseteq \mathcal{G}_2^\infty$. Consider the graph $\mathcal{G}'_{\mathtt{lgg}}$ obtained from $\mathcal{G}_{\mathtt{lgg}}$ by replacing every blank node b by $b_{v_1v_2}$ where $v_1 = \phi_1(b)$ and $v_2 = \phi_2(b)$. Clearly, $\mathcal{G}_{\mathtt{lgg}} \equiv_{\mathcal{R}} \mathcal{G}'_{\mathtt{lgg}}$ holds. Let us show that $\mathcal{G} \models_{\mathcal{R}} \mathcal{G}'_{\mathtt{lgg}}$ holds, i.e., there exists a homomorphism ϕ from the blank nodes in $\mathcal{G}'_{\mathtt{lgg}}$ to the terms in $\mathcal{G}^\infty$ such that $[\mathcal{G}'_{\mathtt{lgg}}]_\phi \subseteq \mathcal{G}^\infty$. By construction of $\mathcal{G}$, it is easy to see that the above holds when ϕ maps $b_{v_1v_2}$ either to v_1 if $v_1 = v_2$ and $v_1 \in \mathcal{U} \cup \mathcal{L}$, or to itself otherwise. It therefore follows that $\mathcal{G} \models_{\mathcal{R}} \mathcal{G}'_{\mathtt{lgg}}$, hence $\mathcal{G} \models_{\mathcal{R}} \mathcal{G}_{\mathtt{lgg}}$. □

As an immediate consequence of the above results, we get the following worst-case bounds for the time to compute an lgg of two RDF graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, and for its size. Here, we assume given the saturation $\mathcal{G}_1^\infty$ and $\mathcal{G}_2^\infty$, as the times to compute them and their sizes depend on the particular set of RDF entailment rules at hand.

COROLLARY 1. *An* lgg *of two RDF graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ can be computed in $O(|\mathcal{G}_1^\infty| \times |\mathcal{G}_2^\infty|)$ and its size is bounded by $|\mathcal{G}_1^\infty| \times |\mathcal{G}_2^\infty|$.*

Figure 4 (right) displays an lgg of the RDF graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ in Figure 3 w.r.t. the entailment rules shown in Table 2. In contrast to Figure 4 (left), which shows an lgg 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 $\mathcal{G}_1$ and i_2 in $\mathcal{G}_2$). Moreover, removing the redundant triples (the grey ones) yields precisely the lgg $\mathcal{G}_{\texttt{lgg}}$ of $\mathcal{G}_1$ and $\mathcal{G}_2$ 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 5. 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 $q_{\texttt{lgg}}$ shown in Figure 5 that asks for the resources having some type. Observe that $q_{\texttt{lgg}}$ is an lgg 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 6 that hold in the scientific publication domain, i.e., the context in which the queries are asked, a more pregnant lgg 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 lgg 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 lgg of BGPQs in Section 4.2.

4.1 Defining the lgg of BGPQs

Recall that the notion of lgg 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 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 lgg *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. Roughly speaking, the saturated query comprises all the triples in the saturation of its body augmented with the constraints, except those triples that are proper to the ontological constraints, i.e., which are not related to what the query is asking for.

DEFINITION 3 (SATURATION W.R.T. CONSTRAINTS). *Given a set $\mathcal{R}$ of RDF entailment rules, the* saturation *of a BGPQ q w.r.t. a set $\mathcal{O}$ of RDFS statements, denoted $q_\mathcal{O}^\infty$, is a BGPQ with the same answer variables as q and whose body, denoted $body(q_\mathcal{O}^\infty)$, is the maximal subset of $(\mathcal{O} \cup body(q))^\infty$ such that for any of its subset $\mathcal{S}$: if $\mathcal{O} \models_\mathcal{R} \mathcal{S}$ holds then $body(q) \models_\mathcal{R} \mathcal{S}$ holds.*

Remark that the above definition coincides with the standard one of saturation (Section 2) when the set $\mathcal{O}$ of ontological constraints is empty.

Figure 6 shows a set $\mathcal{O}$ of ontological constraints, as well as the saturations of the BGPQs q_1 and q_2 in Figure 5 w.r.t. $\mathcal{O}$, named ${q_1}_\mathcal{O}^\infty$ and ${q_2}_\mathcal{O}^\infty$ 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 3. *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 $q_\mathcal{O}^\infty$. For any RDF graph $\mathcal{G}$ whose set of RDFS statements is $\mathcal{O}$,*

1. *$\mathcal{G} \models_\mathcal{R} q$ holds iff $\mathcal{G} \models_\mathcal{R} q_\mathcal{O}^\infty$ holds,*
2. *$q(\mathcal{G}^\infty) = q_\mathcal{O}^\infty(\mathcal{G}^\infty)$ holds.*

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

Item 2) directly follows from: (*i*) the fact that, above, item 1) holds for any homomorphism ϕ and (*ii*) q and $q_\mathcal{O}^\infty$ 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 $q_\mathcal{O}^\infty \models q'$ 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.

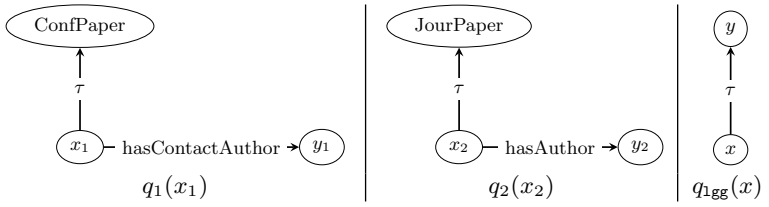

Figure 5: Sample BGPQs q_1, q_2 and their minimal `lgg` $q_{\texttt{lgg}}$.

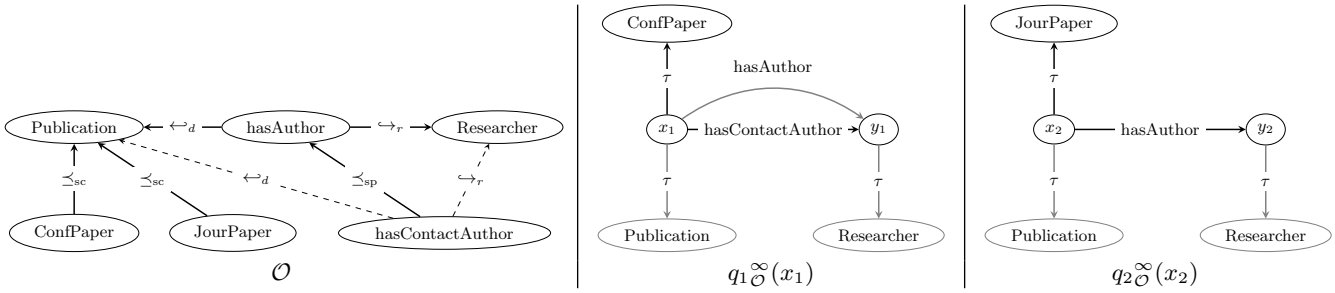

Figure 6: Sample set of ontological constraints $\mathcal{O}$; saturations of the BGPQs q_1 and q_2 in Figure 5 w.r.t. $\mathcal{O}$. Triples shown in grey are redundant w.r.t. those shown in black.

The BGPQ $q(x) \leftarrow (x, \tau, \text{Publication}), (x, \text{hasAuthor}, y)$ is *not* entailed by the queries q_1 and q_2 shown in Figure 5, while it is entailed by these queries w.r.t. the set $\mathcal{O}$ of ontological constraints displayed in Figure 6: $q_1{}^{\infty}_{\mathcal{O}} \models^{\phi_1} q$ (resp. $q_2{}^{\infty}_{\mathcal{O}} \models^{\phi_2} q$) 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 4. *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 $\mathcal{G} \models_{\mathcal{R}} q$ holds then $\mathcal{G} \models_{\mathcal{R}} q'$ holds,*
2. *$q(\mathcal{G}^{\infty}) \subseteq q'(\mathcal{G}^{\infty})$ holds.*

PROOF. Let us first show item 1). By Theorem 3, item 1), $\mathcal{G} \models_{\mathcal{R}} q$ holds iff $\mathcal{G} \models_{\mathcal{R}} q^{\infty}_{\mathcal{O}}$ holds. Consider some ϕ such that $\mathcal{G} \models^{\phi}_{\mathcal{R}} q^{\infty}_{\mathcal{O}}$ holds. Also, by Definition 4, consider some ϕ' such that $q^{\infty}_{\mathcal{O}} \models^{\phi'} q'$. By composing ϕ and ϕ' into ψ, we get $\mathcal{G} \models^{\psi}_{\mathcal{R}} q'$, i.e., $\mathcal{G} \models_{\mathcal{R}} q'$.

Item 2) directly follows from: (i) the fact that, above, item 1) holds for any pair of homomorphisms ϕ, ϕ' such that $\mathcal{G} \models^{\phi}_{\mathcal{R}} q^{\infty}_{\mathcal{O}}$ and $q^{\infty}_{\mathcal{O}} \models^{\phi'} q'$ hold, and (ii) q and $q^{\infty}_{\mathcal{O}}$ 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 containment under deductive constraints* in a database setting [8].

4.1.2 `lgg` 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 `lgg` of queries w.r.t. ontological constraints.

DEFINITION 5 (`lgg` OF BGPQS). *Let $q_1, \ldots, q_n$ be BGPQs with the same arity, $\mathcal{R}$ 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_g such that $q_i \models_{\mathcal{R},\mathcal{O}} q_g$ 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 $q_{\texttt{lgg}}$ of $q_1, \ldots, q_n$ w.r.t. $\mathcal{O}$ such that for any other generalization q_g of $q_1, \ldots, q_n$ w.r.t. $\mathcal{O}$, $q_{\texttt{lgg}} \models_{\mathcal{R},\mathcal{O}} q_g$ holds.*

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

Unlike the case of RDF graphs, an `lgg` of BGPQs may not exist. For instance, consider $q_1(x_1) \leftarrow (x_1, \text{hasAuthor}, y_1)$ 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, there is no BGPQ that generalizes them both.

Similarly to the definition of an `lgg` of RDF graphs, the above definition indicates that an `lgg` of queries is unique up to entailment w.r.t. ontological constraints, i.e., is semantically unique (as $q_{\texttt{lgg}} \models_{\mathcal{R},\mathcal{O}} q_g$ holds for any q_g). Indeed, if it were that queries have *multiples* `lgg`*s incomparable* w.r.t. entailment, say the BGPQs $\texttt{lgg}_1(\bar{x}), \ldots, \texttt{lgg}_m(\bar{x})$, the BGPQ that merges[2] their bodies $q_{\texttt{lgg}}(\bar{x}) \leftarrow \mathit{body}(\texttt{lgg}_1) \uplus$

[2]The *merge* operator of RDF graphs $\uplus$ is straightforwardly extended to union BGPQ bodies (i.e., BGPs) after renaming their blank nodes *and non-answer variables* with fresh ones, so that these BGPs do not share (thus join on) them. Blank nodes and non-answer variables in a BGPQ, which have the *same* semantics (recall Section 2), are used to characterize the answer variables of this BGPQ, hence are *local* to it.

$\cdots \uplus body(\texttt{lgg}_m)$ would be a *single strictly more specific* lgg, a contradiction.

Also, queries may have many syntactically different (though equivalent) lggs due to redundant triples. However, differently from the case of RDF graphs, redundant triples are only those generalizing others in a *purely relation fashion*, as the semantics of BGPQs is *not* its saturation (recall Section 2). This implies that a query has a unique *minimal* lgg, which may be obtained using standard database minimization techniques for relational conjunctive queries. Importantly, redundancy of triples is not specific to lggs of BGPQs, since any query may feature redundancy. We therefore focus in this work on finding *some* lgg of BGPQs.

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

$$\begin{array}{ll} \ell_3(q_1,q_2,q_3) & \equiv_{\mathcal{R},\mathcal{O}} \ \ell_2(\ell_2(q_1,q_2),q_3) \\ \cdots & \cdots \\ \ell_n(q_1,\ldots,q_n) & \equiv_{\mathcal{R},\mathcal{O}} \ \ell_2(\ell_{n-1}(q_1,\ldots,q_{n-1}),q_n) \\ & \equiv_{\mathcal{R},\mathcal{O}} \ \ell_2(\ell_2(\cdots \ell_2(\ell_2(q_1,q_2),q_3)\cdots,q_{n-1}),q_n) \end{array}$$

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. $q_{1\texttt{gg}}$ is an* lgg *of $q_1,\ldots,q_n$ w.r.t. $\mathcal{O}$ iff $q_{1\texttt{gg}}$ is an* lgg *w.r.t. $\mathcal{O}$ of an* lgg *of $q_1,\ldots,q_{n-1}$ and q_n w.r.t. $\mathcal{O}$.*

PROOF. The proof relies on the next lemma.

LEMMA 2. *Let $q_1,\ldots,q_n$ be BGPQs, $\mathcal{O}$ a set of RDFS statements and $\mathcal{R}$ a set of RDF entailment rules. If $q^1_{1\texttt{gg}}$ is an* lgg *of $q_1,\ldots,q_{k<n}$ w.r.t. $\mathcal{O}$ and $q^2_{1\texttt{gg}}$ is an* lgg *of $q_1,\ldots,q_n$ w.r.t. $\mathcal{O}$, then $q^1_{1\texttt{gg}} \models_{\mathcal{R},\mathcal{O}} q^2_{1\texttt{gg}}$ holds.*

Let us show that the above lemma holds.

Suppose that $q^1_{1\texttt{gg}}$ is an lgg of $q_1,\ldots,q_{k<n}$ w.r.t. $\mathcal{O}$, i.e., by Definition 5: (i) $q_{1\leq i\leq k} \models_{\mathcal{R},\mathcal{O}} q^1_{1\texttt{gg}}$ holds and (ii) for any BGPQ q such that $q_{1\leq i\leq k} \models_{\mathcal{R},\mathcal{O}} q$ holds, $q^1_{1\texttt{gg}} \models_{\mathcal{R},\mathcal{O}} q$ holds.

Suppose also that $q^2_{1\texttt{gg}}$ is an lgg of $q_1,\ldots,q_n$ w.r.t. $\mathcal{O}$, i.e., by Definition 5: (i) $q_{1\leq i\leq n} \models_{\mathcal{R},\mathcal{O}} q^2_{1\texttt{gg}}$ holds and (ii) for any BGPQ q' such that $q_{1\leq i\leq n} \models_{\mathcal{R},\mathcal{O}} q'$ holds, $q^2_{1\texttt{gg}} \models_{\mathcal{R},\mathcal{O}} q'$ holds.

Clearly, $q^2_{1\texttt{gg}}$ is a possible value for q above, because $q_{1\leq i\leq n} \models_{\mathcal{R},\mathcal{O}} q^2_{1\texttt{gg}}$ implies $q_{1\leq i\leq k<n} \models_{\mathcal{R},\mathcal{O}} q^2_{1\texttt{gg}}$, hence $q^1_{1\texttt{gg}} \models_{\mathcal{R},\mathcal{O}} q^2_{1\texttt{gg}}$ must hold. $\lhd$

Now, let us prove Proposition 3 using the above lemma.

($\Rightarrow$) Assume that $q_{1\texttt{gg}}$ is an lgg of $q_1,\ldots,q_n$ w.r.t. $\mathcal{O}$ and let us show that it is also an lgg w.r.t. $\mathcal{O}$ of some lgg of $q_1,\ldots,q_{n-1}$ and q_n w.r.t. $\mathcal{O}$. By Definition 5, because $q_{1\texttt{gg}}$ is an lgg of $q_1,\ldots,q_n$ w.r.t. $\mathcal{O}$, we have: (i) $q_{1\leq i\leq n} \models_{\mathcal{R},\mathcal{O}} q_{1\texttt{gg}}$ holds and (ii) for any BGPQ q such that $q_{1\leq i\leq n} \models_{\mathcal{R},\mathcal{O}} q$ holds, $q_{1\texttt{gg}} \models_{\mathcal{R},\mathcal{O}} q$ holds.

Let $q'_{1\texttt{gg}}$ be an lgg of $q_1,\ldots,q_{n-1}$ w.r.t. $\mathcal{O}$. From (i) and the above lemma, (*) $q'_{1\texttt{gg}} \models_{\mathcal{R},\mathcal{O}} q_{1\texttt{gg}}$ and $q_n \models_{\mathcal{R},\mathcal{O}} q_{1\texttt{gg}}$ hold. Moreover, for any BGPQ q, if it were that $q'_{1\texttt{gg}} \models_{\mathcal{R},\mathcal{O}} q$ and $q_n \models_{\mathcal{R},\mathcal{O}} q$ and $q_{1\texttt{gg}} \not\models_{\mathcal{R},\mathcal{O}} q$, then $q_{1\leq i\leq n} \models_{\mathcal{R},\mathcal{O}} q$ would hold and contradict the fact that $q_{1\texttt{gg}}$ is an lgg of $q_1,\ldots,q_n$ w.r.t. $\mathcal{O}$. Therefore, (**) for any BGPQ q, if $q'_{1\texttt{gg}} \models_{\mathcal{R},\mathcal{O}} q$ and $q_n \models_{\mathcal{R},\mathcal{O}} q$ holds, then $q_{1\texttt{gg}} \models_{\mathcal{R},\mathcal{O}} q$ holds.

From Definition 5, (*) and (**) we get that $q_{1\texttt{gg}}$ is an lgg w.r.t. $\mathcal{O}$ of an lgg of $q_1,\ldots,q_{n-1}$ and q_n w.r.t. $\mathcal{O}$.

($\Leftarrow$) Assume that $q_{1\texttt{gg}}$ is an lgg w.r.t. $\mathcal{O}$ of an lgg q' of $q_1,\ldots,q_{n-1}$ and q_n w.r.t. $\mathcal{O}$, and let us show that it is also an lgg of $q_1,\ldots,q_n$ w.r.t. $\mathcal{O}$.

By Definition 5, (i) $q' \models_{\mathcal{R},\mathcal{O}} q_{1\texttt{gg}}$ and $q_n \models_{\mathcal{R},\mathcal{O}} q_{1\texttt{gg}}$ hold, hence $q_{1\leq i\leq n-1} \models_{\mathcal{R},\mathcal{O}} q_{1\texttt{gg}}$ and $q_n \models_{\mathcal{R},\mathcal{O}} q_{1\texttt{gg}}$ hold, i.e., (*') $q_{1\leq i\leq n} \models_{\mathcal{R},\mathcal{O}} q_{1\texttt{gg}}$ holds.

Moreover, (ii) for any BGPQ q such that $q' \models_{\mathcal{R},\mathcal{O}} q$ and $q_n \models_{\mathcal{R},\mathcal{O}} q$ hold, $q_{1\texttt{gg}} \models_{\mathcal{R},\mathcal{O}} q$ holds. By Definition 5, $q_{1\leq i\leq n-1} \models_{\mathcal{R},\mathcal{O}} q'$ holds, therefore we get: (**') for any BGPQ q such that $q_{1\leq i\leq n} \models_{\mathcal{R},\mathcal{O}} q$ holds, $q_{1\texttt{gg}} \models_{\mathcal{R},\mathcal{O}} q$ holds.

From Definition 5, (*') and (**') we get that $q_{1\texttt{gg}}$ is an lgg of $q_1,\ldots,q_n$ w.r.t. $\mathcal{O}$. $\square$

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 lgg *of q_1 and q_2 w.r.t. $\mathcal{O}$.*

4.2 Computing an 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 5) that it corresponds to an lgg of q_1 and q_2 just in case extra ontological constraints and RDF entailment are ignored. Then, we show (Theorem 6) that their enhanced 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 at hand (Definition 3). We also provide the size of these cover query-based lggs, as well as the time to compute them.

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

- *$(t_1,t_2,t_3) \in body(q_1)$ and $(t_4,t_5,t_6) \in body(q_2)$ iff $(t_7,t_8,t_9) \in body(q)$ with, for $1 \leq i \leq 3$, $t_{i+6} = t_i$ if $t_i = t_{i+3}$ and $t_i \in \mathcal{U} \cup \mathcal{L}$, otherwise t_{i+6} is the variable $v_{t_i t_{i+3}}$*
- *$head(q) = q(v_{x^1_1 x^1_2},\ldots,v_{x^n_1 x^n_2})$ iff $head(q_1) = q(x^1_1,\ldots,x^n_1)$ and $head(q_2) = q(x^1_2,\ldots,x^n_2)$*

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 1), so that it corresponds to an 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 BGPs, which generalize RDF graphs by allowing variables in triples (Section 2). In particular, unknown 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 1 from RDF graphs to BGPs, (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

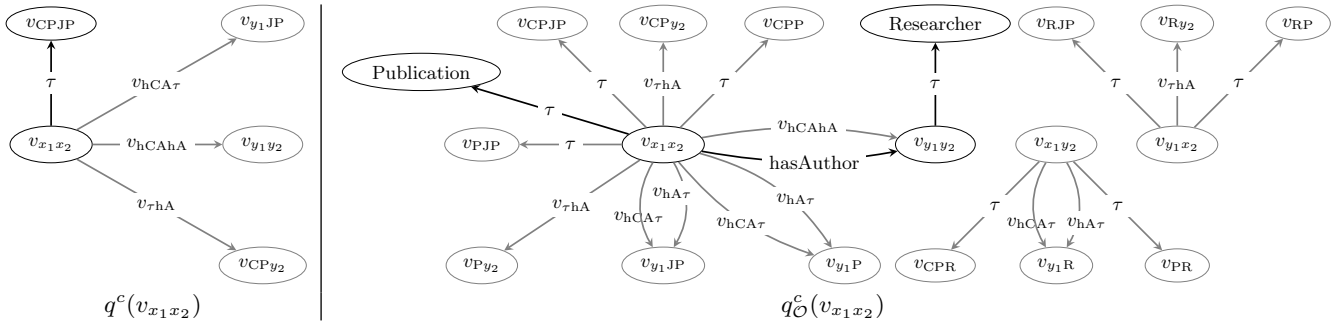

Figure 7: Cover queries of the BGPQs q_1 and q_2 in Figure 5 (left), and of their saturations w.r.t. $\mathcal{O}$ in Figure 6 (right). Triples shown in grey are redundant w.r.t. those shown in black.

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 7 (left) shows the cover query q^c of the BGPQs q_1 and q_2 displayed in Figure 5. The head $q^c(v_{x_1x_2})$ of q^c results from anti-unifying $q_1(x_2)$ and $q_2(x_2)$; its body consists of the triple $(v_{x_1x_2}, \tau, v_{\text{CPJP}})$, which results from anti-unifying $(x_1, \tau, \text{ConfPaper})$ in q_1 and $(x_2, \tau, \text{JourPaper})$ in q_2, of the triple $(v_{x_1x_2}, v_{\text{hCA}\tau}, v_{y_1\text{JP}})$, which results from anti-unifying $(x_1, \text{hasContactAuthor}, y_1)$ in q_1 and $(x_2, \tau, \text{JourPaper})$ in q_2 etc.

Importantly, a cover query of two BGPQs may not exist (If ... then ... 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 `lgg` may no exists. Their cover query does not exists either, since Definition 6 leads to $q(v_{x_1x_2}) \leftarrow (v_{x_1y_2}, \text{hasAuthor}, v_{y_1x_2})$ which is *not* a BGPQ ($v_{x_1x_2}$ does not appear in q's body).

The next theorem formalizes the above discussion:

THEOREM 5. *Given two BGPQs q_1, q_2 with the same arity and empty sets $\mathcal{R}$ of RDF entailment rules and $\mathcal{O}$ of RDFS statements:*

1. *the cover query of q_1 and q_2 exists iff an* `lgg` *of q_1 and q_2 exists;*
2. *the cover query of q_1 and q_2 is an* `lgg` *of q_1 and q_2.*

PROOF. The proof can be directly derived from that of the more general Theorem 6 (see below), noting that in the particular case of Theorem 5 $\mathcal{R} = \emptyset$ and $\mathcal{O} = \emptyset$ hold, hence ${q_1}^\infty_\mathcal{O} = q_1$ and ${q_2}^\infty_\mathcal{O} = q_2$ hold. □

In Figure 7 (left), the cover query q^c of the BGPQs q_1 and q_2 displayed in Figure 5 is therefore an `lgg` of them when both extra ontological constraints and RDF entailment are ignored (Theorem 5). Remark that q^c is equivalent to the naïve minimal `lgg` of q_1 and q_2 shown Figure 5: q_{lgg}.

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 $O(|body(q_1)| \times |body(q_2)|)$; its size is $|body(q_1)| \times |body(q_2)|$.*

The next theorem generalizes the preceding one in order to use the notion of cover query to compute an `lgg` of two queries *w.r.t. extra ontological constraints* and *any* set of RDF entailment rules.

THEOREM 6. *Given a set $\mathcal{R}$ 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 q of ${q_1}^\infty_\mathcal{O}$ and ${q_2}^\infty_\mathcal{O}$ exists iff an* `lgg` *of q_1 and q_2 w.r.t. $\mathcal{O}$ exists;*
2. *the cover query q of ${q_1}^\infty_\mathcal{O}$ and ${q_2}^\infty_\mathcal{O}$ is an* `lgg` *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., $q_1 \models_{\mathcal{R},\mathcal{O}} q$ *and* that $q_2 \models_{\mathcal{R},\mathcal{O}} q$. Clearly, once proved, this entails de fact that if q exists, then an `lgg` of q_1 and q_2 w.r.t. $\mathcal{O}$ exists.

Consider the BGPQ q' obtained from q by replacing every variable $v_{t_1t_2}$ by the value t_1. Clearly, $q' \models_{\mathcal{R},\mathcal{O}} q$ and, by construction of q, $q' = q_1$, i.e., $q_1 \models_{\mathcal{R},\mathcal{O}} q$. Showing that $q_2 \models_{\mathcal{R},\mathcal{O}} q$ 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 `lgg` 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 `lgg` of q_1 and q_2 w.r.t. $\mathcal{O}$. Let q_{lgg} be *any* `lgg` 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). Also, assume that $q, q_1, q_2, q_{\text{lgg}}$ heads are $q(\bar{x}), q_1(\bar{x_1}), q_2(\bar{x_2}), q_{\text{lgg}}(\bar{x}_{\text{lgg}})$ respectively.

To prove our claim, we need to show that $q \models_{\mathcal{R},\mathcal{O}} q_{\text{lgg}}$ holds. Since q_{lgg} is an `lgg` of q_1 and q_2 w.r.t. $\mathcal{O}$, there exist two homomorphisms ϕ_1 and ϕ_2 from the variables in q_{lgg} to the values (variables and constants) in ${q_1}^\infty_\mathcal{O}$ and in ${q_2}^\infty_\mathcal{O}$ respectively, such that $[body(q_{\text{lgg}})]_{\phi_1} \subseteq body({q_1}^\infty_\mathcal{O})$ and $\phi_1(\bar{x}_{\text{lgg}}) = \bar{x_1}$, and similarly, $[body(q_{\text{lgg}})]_{\phi_2} \subseteq body({q_2}^\infty_\mathcal{O})$ and $\phi_2(\bar{x}_{\text{lgg}}) = \bar{x_2}$. Consider the BGPQ q'_{lgg} obtained from q_{lgg} by replacing every variable v by $v_{t_1t_2}$ where $t_1 = \phi_1(v)$ and

$t_2 = \phi_2(v)$. Clearly, $q_{\texttt{lgg}} \equiv_{\mathcal{R},\mathcal{O}} q'_{\texttt{lgg}}$ holds. Assume that the head of $q'_{\texttt{lgg}}$ is $q'_{\texttt{lgg}}(\bar{x}'_{\texttt{lgg}})$; by construction is has the same head as q. Let us show that $q \models_{\mathcal{R},\mathcal{O}} q'_{\texttt{lgg}}$ holds, i.e., there exists a homomorphism ϕ from the variables in $q'_{\texttt{lgg}}$ to values in $q^{\infty}_{\mathcal{O}}$ such that $[body(q'_{\texttt{lgg}})]_{\phi} \subseteq body(q^{\infty}_{\mathcal{O}})$ and $\phi(x'_{\texttt{lgg}}) = \bar{x}$. By Definition of q, it is easy to see that the above holds when ϕ maps $v_{t_1 t_2}$ either to t_1 if $t_1 = t_2$ and $t_1 \in \mathcal{U} \cup \mathcal{L}$, or to itself otherwise. Importantly, this entails (i) the existence of the cover query q when an `lgg` $q_{\texttt{lgg}}$ exists and (ii) that $q \models_{\mathcal{R},\mathcal{O}} q'_{\texttt{lgg}}$ holds, hence $q \models_{\mathcal{R},\mathcal{O}} q_{\texttt{lgg}}$ holds, and therefore q is an `lgg` 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 `lgg` of two BGPQs q_1 and q_2, and its size. We assume given the saturation ${q_1}^{\infty}_{\mathcal{O}}$ and ${q_2}^{\infty}_{\mathcal{O}}$ w.r.t. the sets $\mathcal{O}$ of ontological constraints and $\mathcal{R}$ of RDF entailment rules under consideration, as the times to compute ${q_1}^{\infty}_{\mathcal{O}}$ and ${q_2}^{\infty}_{\mathcal{O}}$, and their sizes, depend on the particular $\mathcal{O}$ and $\mathcal{R}$ at hand.

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

Figure 7 (right) displays the cover query of the BGPQs ${q_1}^{\infty}_{\mathcal{O}}$ and ${q_2}^{\infty}_{\mathcal{O}}$ shown in Figure 6. It is therefore (Theorem 6) an `lgg` of the BGPQs q_1 and q_2 shown in Figure 5 w.r.t. the ontological constraints shown in Figure 6, using the RDF entailment rules shown in Table 2.

Figure 7 exemplifies the benefits of taking into 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. EXPERIMENTS

We provide an experimental assessment of our technical contribution for computing `lgg`s of BGPQs, which goes beyond that for RDF graphs (recall that BGPQ bodies generalize RDF graphs). In Section 5.1, we describe our experimental settings. Then, in Section 5.2, we study the saturation of a BGPQ w.r.t. ontological constraints. Finally, in Section 5.3, we study the computation of `lgg`s with or without considering constraints, notably the gain in precision when constraints are considered.

5.1 Settings

Software. We implemented our framework for computing `lgg`s of RDF graphs or of BGPQs in Java 1.8 (https://www.java.com), on top of the Jena 3.0.1 RDF reasoner (https://jena.apache.org) and a PostgreSQL 9.3.11 server (https://www.postgresql.org); all used with default settings.

We use Jena to compute the saturation of an RDF graph (against which queries must be evaluated to obtained their complete answer sets, recall Section 2), as well as the saturation of queries w.r.t. extra ontological constraints that we devised in the preceding section.

PostgreSQL is used to evaluate *SQLized* BGPQs against a saturated RDF graph. This RDF graph, which comprises both data and constraints (i.e., RDF and RDFS statements), is stored in a `Triple(s,p,o)` PostgreSQL table, indexed by all permutations of the `s`, `p`, `o` columns, leading to a total of 6 indexes. This indexing choice is inspired by [17, 28], to give PostgreSQL efficient query evaluation opportunities.

There exist alternative data layouts like having one two-columns table `p(s,o)` per distinct property `p` in the RDF graph, which stores all the subject/object tuples of triples with property `p` [6], or the elaborate layout of [4] 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 rules shown in Table 2, which allow reasoning about RDFS ontological constraints.

Dataset. We conducted experiments using LUBM [13]. We generated an RDF graph comprising 884k triples (including 242 ontological constraints, i.e., RDFS statements) before saturation and 1.08M triples after. This RDF graph is used to compare the quality of `lgg`s of BGPQs w.r.t. their precision, when ontological are considered or not.

Queries. We borrowed from [5] a set of 9 BGPQs. Table 3 (top) displays the characteristics of these queries, which have a variety of structural aspects and numbers of answers. The queries can be found in the Appendix.

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.

5.2 BGPQ saturation w.r.t. constraints

Following Definition 3, we compute the saturation of a BGPQ q w.r.t. a set $\mathcal{O}$ of ontological constraints as follows. We build a saturated query $q^{\infty}_{\mathcal{O}}$ with same output variables as q and whose body comprises the triples in $(body(q) \cup \mathcal{O})^{\infty}$, except those in $\mathcal{O}^{\infty}$ that are not in $body(q)^{\infty}$, i.e., $body(q^{\infty}_{\mathcal{O}}) = (body(q) \cup \mathcal{O})^{\infty} \setminus (\mathcal{O}^{\infty} \setminus body(q)^{\infty})$.

Table 3 (bottom) shows the size of our saturated test queries and the time to compute them. Enriching our test queries using the 242 LUBM constraints significantly augments their size: from $\times 2$ for Q_{10} up to $\times 5$ for Q_{22}. The query saturation time is always fast: a very few tens of milliseconds for all our test queries.

5.3 `lgg`s of BGPQs w.r.t. constraints

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).

Further, observe that the cover query-based `lgg` of two BGPQs when ignoring extra ontological constraints, is *entailed* by that of these same queries when extra contraints are considered. Indeed, by construction, these `lgg`s have the same answer variables and, clearly, the body of the former is included in that of the latter (as it is built using the saturated queries an not the queries themselves). This observation allows comparing their numbers of answers in order to *quantify* to which extent `lgg`s ignoring extra ontological

Queries:	Q_{01}	Q_{03}	Q_{05}	Q_{07}	Q_{08}	Q_{10}	Q_{12}	Q_{22}	Q_{25}
shape	star	star	graph	graph	star	tree	graph	star	star
size (number of triples)	3	3	4	3	2	4	6	2	3
number of constants/distinct variables/answer variables	5/2/2	5/2/2	5/3/3	4/2/2	3/2/2	6/3/3	7/4/3	2/3/2	5/2/2
cardinality (number of answers)	123	41	869	0	269	41 751	79	14 252	16
size of the saturation w.r.t. constraints	8	9	11	7	6	8	14	10	8
time to compute the saturation w.r.t. constraints	24	23	23	21	22	22	21	27	22

Table 3: Characteristics of our test queries (top) and of their saturations w.r.t. LUBM constraints (bottom); times are in ms.

Pairs of queries:	$Q_{01}Q_{07}$	$Q_{03}Q_{07}$	$Q_{01}Q_{22}$	$Q_{03}Q_{22}$	$Q_{08}Q_{22}$	$Q_{01}Q_{25}$	$Q_{22}Q_{25}$	$Q_{05}Q_{10}$	$Q_{05}Q_{12}$
time to compute the `lgg`	1	2	1	1	1	3	2	2	4
cardinality (number of answers) of the `lgg`	30 963	1 048 060	1 048 060	74 643	1 048 060	253 443	1 048 060	9 615 770	34 852
time to compute the `lgg` w.r.t. constraints	5	6	6	7	10	6	6	6	8
cardinality of the `lgg` w.r.t. constraints	14 285	14 285	33 305	33 305	340 358	74 643	33 305	351 580	30 529

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

constraints are unnecessarily more general than `lgg`s which take them into account, i.e., to which extent `lgg`s considering extra ontological constraints are more *precise* than those ignoring such constraints. Table 4 (lines 2 and 4) shows that ignoring extra constraints significantly increases the number of answers for some `lgg`s, from a small $\times 1.14$ for $Q_{05}Q_{12}$ up to a striking $\times 73$ for $Q_{03}Q_{07}$, with a significant average of $\times 19.42$. `lgg`s with same number of answers in Table 4 were found equivalent, e.g., `lgg`s with 1 048 060 answers are asking for all the (distinct) subject/object pairs in triples in the LUBM RDF graph.

6. RELATED WORK

The problem of computing an `lgg` was introduced in the early 70's by G. Plotkin [21, 22] to generalize First Order Logic clauses w.r.t. θ-subsumption, a non-standard logical implication typical of Machine Learning.

Recently, this problem has started receiving consideration in the Semantic Web field:

[10] studies the problem of computing an `lgg` of two so-called *r-graphs* while *ignoring RDF entailment rules*, i.e., comparing r-graphs in a purely relational fashion (recall Section 2). r-graphs are *particular* RDF graphs describing a *single* resource r called the root. More precisely, r-graphs are *almost-tree* RDF graphs, which can be represented as standard trees if we consider as pattern of successive edges: a triple $(\mathtt{s}_1, \mathtt{p}_1, \mathtt{o}_1)$ followed by either (i) $(\mathtt{o}_1, \mathtt{p}_2, \mathtt{o}_2)$ or (ii) $(\mathtt{p}_1, \mathtt{p}_2, \mathtt{o}_2)$. Intuitively, an r-graph captures the fact that r is described by the resources it is (indirectly) connected to (i) *and* by the properties (indirectly) connecting it to other resources (ii). The proposed technique for computing an `lgg` of two r-graphs exploits their tree-shapes to traverse them simultaneously from the roots up to the leaves, while constructing an r-graph `lgg`. Our contributions encompass those of [10], since we can compute an `lgg` of general RDF graphs using any set of RDF entailment rules.

[15] investigates the problem of computing an `lgg` of *unary tree-BGPQs* while *ignoring RDF entailment rules*. Intuitively, a unary tree BGPQ describes its answer variable by how it must be (indirectly) connected to other resources. Like in [10], a tree BGPQ `lgg` is computed by a simultaneous root-to-leaves traversal of the tree shaped input queries. Our contributions encompass those of [15], since we can compute an `lgg` of general n-ary BGPQs using any set of RDF entailment rules and, further, extra ontological constraints.

The problem of computing an `lgg` has also been studied in Knowledge Representation, where the notion of `lgg` was rebaptized *least common subsumer*. Interestingly, some works of this field use formalisms whose expressivity overlaps with our RDF/SPARQL setting.

In Description Logics (DLs), [2] studies the problem of computing an `lgg` of concepts (DL formulae) for three DLs: $\mathcal{EL}$ and it extensions $\mathcal{FLE}$ and $\mathcal{ALE}$; these results are extended in [3, 29] to consider ontological constraints. $\mathcal{EL}$-concepts correspond exactly to *unary tree BGPQs in which all triple properties are constants*, while the only shared constraints are *domain typing and subclass*. For these $\mathcal{EL}$-concepts and particular BGPQs, one may interchangeably use the $\mathcal{EL}$-technique or ours to compute `lgg`s.

Similarly, in Conceptual Graphs (CGs), the so-called *simple CGs with unary and binary relations* [9] correspond to *RDF graphs in which all triples with τ property must have a constant has class name*; *all the RDFS ontological constraints* are shared with simple CGs. For these simple CGs and particular RDF graphs, one may interchangeably use the CG-technique or ours to compute `lgg`s.

7. 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 *almost-tree* 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 de-

scribing the application domain in which queries are posed may enhance the precision of lggs.

As a short-term perspective, we want to study heuristics in order to efficiently prune out as much as possible redundant triples, while computing lggs. Indeed, as for instance Figures 4 and 7 show, our cover graph/query technique produces many redundant triples. This would allow having more readable lggs, 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 improves their evaluation time.

Computing lggs 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 $\mathcal{V}$ of lggs of queries from a workload, such that (*i*) the workload queries can be (partially or totally) rewritten using $\mathcal{V}$ and (*ii*) $\mathcal{V}$ minimizes a combination of query processing, view storage, and view maintenance costs. This would provide an alternative to [11], 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 lggs, and apply it to view selection, in the setting of the DL-Lite$_{\mathcal{R}}$ description logic [7], 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).

8. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] F. Baader, R. Kiisters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *IJCAI*, 1999.

[3] F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logic*, 5(3), 2007.

[4] M. A. Bornea, J. Dolby, A. Kementsietsidis, K. Srinivas, P. Dantressangle, O. Udrea, and B. Bhattacharjee. Building an efficient RDF store over a relational database. In *SIGMOD*, 2013.

[5] D. Bursztyn, F. Goasdoué, and I. Manolescu. Optimizing reformulation-based query answering in RDF. In *EDBT*, 2015.

[6] D. Bursztyn, F. Goasdoué, and I. Manolescu. Teaching an RDBMS about ontological constraints. *PVLDB*, 9(12), 2016.

[7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3), 2007.

[8] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *PODS*, 1998.

[9] M. Chein and M. Mugnier. *Graph-based Knowledge Representation - Computational Foundations of Conceptual Graphs*. Springer, 2009.

[10] S. Colucci, F. Donini, S. Giannini, and E. D. Sciascio. Defining and computing least common subsumers in RDF. *J. Web Semantics*, 39(0), 2016.

[11] F. Goasdoué, K. Karanasos, J. Leblay, and I. Manolescu. View selection in semantic web databases. *PVLDB*, 5(2), 2011.

[12] F. Goasdoué, I. Manolescu, and A. Roatiş. Efficient query answering against dynamic RDF databases. In *EDBT*, 2013.

[13] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics*, 3(2-3), Oct. 2005.

[14] T. Imielinski and W. Lipski. Incomplete information in relational databases. *JACM*, 31(4), 1984.

[15] J. Lehmann and L. Bühmann. Autosparql: Let users query your knowledge base. In *ESWC*, 2011.

[16] M. Meier. Towards rule-based minimization of RDF graphs under constraints. In *Web Reasoning and Rule Systems*, 2008.

[17] T. Neumann and G. Weikum. x-rdf-3x: Fast querying, high update rates, and consistency for RDF databases. *PVLDB*, 3(1), 2010.

[18] F. Picalausa, Y. Luo, G. H. Fletcher, J. Hidders, and S. Vansummeren. A structural approach to indexing triples. In *ESWC*, 2012.

[19] R. Pichler, A. Polleres, S. Skritek, and S. Woltran. Redundancy elimination on RDF graphs in the presence of rules, constraints, and queries. In *Web Reasoning and Rule Systems*, 2010.

[20] R. Pichler, A. Polleres, S. Skritek, and S. Woltran. Complexity of redundancy detection on RDF graphs in the presence of rules, constraints, and queries. *Semantic Web*, 4(4), 2013.

[21] G. D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5, 1970.

[22] G. D. Plotkin. A further note on inductive generalization. *Machine Intelligence*, 6, 1971.

[23] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1), Jan. 1965.

[24] J. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning*. Elsevier and MIT Press, 2001.

[25] Resource description framework 1.1. https://www.w3.org/TR/rdf11-concepts.

[26] RDF 1.1 semantics. https://www.w3.org/TR/rdf11-mt/.

[27] SPARQL protocol and RDF query language. http://www.w3.org/TR/rdf-sparql-query.

[28] C. Weiss, P. Karras, and A. Bernstein. Hexastore: sextuple indexing for semantic web data management. *PVLDB*, 1(1), 2008.

[29] B. Zarrieß and A. Turhan. Most specific generalizations w.r.t. general EL-TBoxes. In *IJCAI*, 2013.

9. APPENDIX

$Q01(?X, ?Y) : -$ $?X$ "$http://www.w3.org/1999/02/22\text{-}rdf\text{-}syntax\text{-}ns\#type$" "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#Employee$", $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#worksFor$" "$http://www.Department0.University0.edu$", $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#degreeFrom$" $?Y$
$Q03(?X, ?Y) : -$ $?X$ "$http://www.w3.org/1999/02/22\text{-}rdf\text{-}syntax\text{-}ns\#type$" "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#Employee$", $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#worksFor$" "$http://www.Department0.University0.edu$", $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#doctoralDegreeFrom$" $?Y$
$Q05(?X, ?Y, ?Z) : -$ $?X$ "$http://www.w3.org/1999/02/22\text{-}rdf\text{-}syntax\text{-}ns\#type$" "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#Student$", $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#advisor$" $?Y$, $?Y$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#teacherOf$" $?Z$, $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#takesCourse$" $?Z$,
$Q07(?X, ?Y) : -$ $?X$ "$http://www.w3.org/1999/02/22\text{-}rdf\text{-}syntax\text{-}ns\#type$" "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#Faculty$", $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#degreeFrom$" $?Y$, $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#memberOf$" $?Y$
$Q08(?X, ?Y) : -$ $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#degreeFrom$" $?Y$, $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#memberOf$" "$http://www.Department0.University0.edu$",
$Q10(?W, ?X, ?Y) : -$ $?X$ "$http://www.w3.org/1999/02/22\text{-}rdf\text{-}syntax\text{-}ns\#type$" "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#GraduateStudent$", $?Y$ "$http://www.w3.org/1999/02/22\text{-}rdf\text{-}syntax\text{-}ns\#type$" "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#Faculty$", $?W$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#publicationAuthor$" $?X$, $?W$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#publicationAuthor$" $?Y$
$Q12(?W, ?X, ?Y) : -$ $?X$ "$http://www.w3.org/1999/02/22\text{-}rdf\text{-}syntax\text{-}ns\#type$" "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#GraduateStudent$", $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#advisor$" $?Y$, $?Y$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#teacherOf$" $?Z$, $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#takesCourse$" $?Z$, $?W$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#publicationAuthor$" $?X$, $?W$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#publicationAuthor$" $?Y$
$Q22(?X, ?Y) : -$ $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#doctoralDegreeFrom$" $?Z$, $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#teacherOf$" $?Y$
$Q25(?X, ?Y) : -$ $?X$ "$http://www.w3.org/1999/02/22\text{-}rdf\text{-}syntax\text{-}ns\#type$" "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#Faculty$", $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#degreeFrom$" "$http://www.University532.edu$", $?X$ "$http://swat.cse.lehigh.edu/onto/univ\text{-}bench.owl\#memberOf$" $?Y$

Figure 8: LUBM queries