Learning Commonalities in SPARQL

Sara El Hassad, François Goasdoué(✉), and Hélène Jaudoin

IRISA, Univ. Rennes 1, Lannion, France
{sara.el-hassad,fg,helene.jaudoin}@irisa.fr

Abstract. Finding the commonalities between descriptions of data or knowledge is a foundational reasoning problem of Machine Learning. It was formalized in the early 70's as computing a *least general generalization (*lgg*)* of such descriptions. We revisit this well-established problem in the SPARQL query language for RDF graphs. In particular, and by contrast to the literature, we address it for the *entire* class of conjunctive SPARQL queries, a.k.a. Basic Graph Pattern Queries (BGPQs), and crucially, when *background knowledge* is available as RDF Schema ontological constraints, we take advantage of it to devise much more precise lggs, as our experiments on the popular DBpedia dataset show.

Keywords: BGP queries · RDF · RDFS · Least general generalization

1 Introduction

Finding the commonalities between descriptions of data or knowledge is a fundamental Machine Learning problem, which was formalized in the early 70's as computing a *least general generalization* (lgg) of such descriptions [21]. Since then, it has also received consideration in the Knowledge Representation field, where least general generalizations were rebaptized *least common subsumers* [5], in Description Logics [1,5,14,27] and in Conceptual Graphs [3]. More recently, this problem started being investigated in RDF [7,9] and its associated SPARQL query language [2,10,16], the two prominent Semantic Web standards by W3C.

Motivations. We study this old reasoning problem in the *SPARQL* setting (contributions to be outlined shortly), i.e., when input descriptions are SPARQL queries. Solutions to this problem can be applied to a variety of useful important applications, ranging from *optimization* to *exploration* and *recommendation* in RDF data management systems or in SPARQL endpoints. For instance, an lgg of incoming queries characterizes the largest set of their commonalities whose processing may be shared in *multi-query optimization* [15]. Similarly, lggs of subsets of a query workload correspond to candidate views that may be recommended for materialization in *view selection* [11], a typical optimization for data warehouses [6], and among which can be selected those that allow rewriting (partially or totally) the workload while minimizing a combination of rewriting processing, view storage and view maintenance costs. Also, *clustering user queries* found in system logs, based on their lggs, may help classifying the

C. d'Amato et al. (Eds.): ISWC 2017, Part I, LNCS 10587, pp. 278–295, 2017.
DOI: 10.1007/978-3-319-68288-4_17

queries and identifying the kind of data each category accesses [4]. Finally, finding the relevant user query cluster for an incoming query may help recommending *similar and complementary searches* [13].

Contributions. We bring the following contributions to the problem of finding an `lgg` of SPARQL queries:

1. We carefully study a novel notion of `lgg` for the popular conjunctive fragment of SPARQL (Sect. 3), a.k.a. Basic Graph Pattern Queries (BGPQs). Our definition, which we briefly outlined in [10], significantly departs from the literature by (*i*) considering *general* BGPQs, instead of *unary tree-shaped* ones [2,16], and crucially by (*ii*) taking advantage of *background knowledge* formalized as RDF Schema (RDFS) ontological constraints. Furthermore, to establish this definition of an `lgg`, we revise the standard generalization/specialization relation (a.k.a. entailment) between BGPQs in order to devise a *well-founded* entailment relation that allows comparing BGPQs *w.r.t. extra RDFS constraints*, i.e., the counterpart to *subsumption between concepts w.r.t. a terminology* in Description Logics and to *containment between queries w.r.t. constraints* in Databases.
2. We provide a solution to the above problem (Sect. 4), which technically differs from the state of the art [2,16] in that it cannot exploit the (imposed) tree-shape of the input BGPQs to compute their `lgg` through a simultaneous root-to-leaves traversal. Instead, our solution traverses blindly the general (hence arbitrary-shaped) input BGPQs and builds their `lgg` using the notion of *least general anti-unification* of atoms [21,23], which is dual to the well-known notion of most general unification of atoms [22,23]. Also, to take into account background knowledge, we define a *well-founded* notion of *saturation of BGPQs w.r.t. extra RDFS constraints*, which we devise inspired by that of RDF graphs.
3. We report on experiments made to assess the added-value of considering background knowledge when computing `lgg`s of BGPQs (Sect. 5). Notably, we use real data from DBpedia to show how much more precise `lgg`s are when background knowledge is considered, by measuring the *gain in precision* it yields.

Organization. Following the presentation of [8–10], we first recall the basics of RDF and SPARQL in Sect. 2. Then, we detail the above contributions. Finally, we discuss related work and conclude in Sect. 6.

Supplementary material (proofs of our technical results, implemented algorithms, additional experiments, etc) is available in our online research report [8].

2 Preliminaries

2.1 The Resource Description Framework (RDF)

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* `s`

Table 1. RDF & RDFS statements.

RDF statement	Triple
Class assertion	$(\mathtt{s}, \text{rdf:type}, \mathtt{o})$
Property assertion	$(\mathtt{s}, \mathtt{p}, \mathtt{o})$ with $\mathtt{p} \neq \text{rdf:type}$
RDFS statement	Triple
Subclass	$(\mathtt{s}, \text{rdfs:subClassOf}, \mathtt{o})$
Subproperty	$(\mathtt{s}, \text{rdfs:subPropertyOf}, \mathtt{o})$
Domain typing	$(\mathtt{s}, \text{rdfs:domain}, \mathtt{o})$
Range typing	$(\mathtt{s}, \text{rdfs:range}, \mathtt{o})$

Table 2. Sample RDF entailment rules.

Rule [25]	Entailment rule
rdfs2	$(\mathtt{p}, \hookleftarrow_d, \mathtt{o}), (\mathtt{s}_1, \mathtt{p}, \mathtt{o}_1) \rightarrow (\mathtt{s}_1, \tau, \mathtt{o})$
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	$(\mathtt{p}, \hookleftarrow_d, \mathtt{o}), (\mathtt{o}, \preceq_{\text{sc}}, \mathtt{o}_1) \rightarrow (\mathtt{p}, \hookleftarrow_d, \mathtt{o}_1)$
ext2	$(\mathtt{p}, \hookrightarrow_r, \mathtt{o}), (\mathtt{o}, \preceq_{\text{sc}}, \mathtt{o}_1) \rightarrow (\mathtt{p}, \hookrightarrow_r, \mathtt{o}_1)$
ext3	$(\mathtt{p}, \preceq_{\text{sp}}, \mathtt{p}_1), (\mathtt{p}_1, \hookleftarrow_d, \mathtt{o}) \rightarrow (\mathtt{p}, \hookleftarrow_d, \mathtt{o})$
ext4	$(\mathtt{p}, \preceq_{\text{sp}}, \mathtt{p}_1), (\mathtt{p}_1, \hookrightarrow_r, \mathtt{o}) \rightarrow (\mathtt{p}, \hookrightarrow_r, \mathtt{o})$

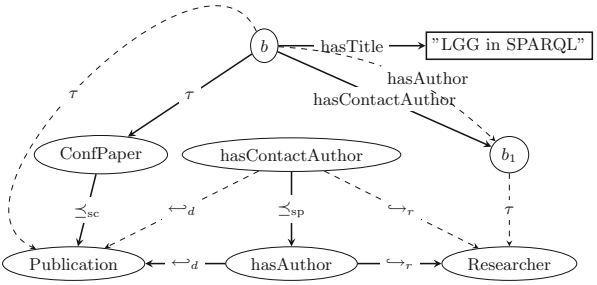

Fig. 1. Sample RDF graph $\mathcal{G}$.

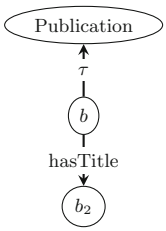

Fig. 2. Sample RDF graph $\mathcal{G}'$.

has the *property* p, the value of which is the *object* 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 are identifiers for missing values (unknown URIs or literals). *Well-formed triples*, as per the RDF specification [24], 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 s, p, 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 $(b, \text{hasTitle})$, "LGG in SPARQL" and $(b, \text{hasContactAuthor}, b_1)$ mean: *something (b) entitled "LGG in SPARQL" 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. The RDF standard [24] 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 $(b, \text{rdf:type}, \text{ConfPaper})$.

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 built-in properties. Table 1 (bottom) lists the allowed constraints and the triples to state them; *domain* and *range* denote respectively the first and second attribute of every property. For example, the triple (ConfPaper, rdfs:subClassOf, Publication) states that *conference papers are publications*, the triple (hasContactAuthor, rdfs:subPropertyOf, hasAuthor) states that *having a contact author is having an author*, the triple (hasAuthor, rdfs:domain, Publication) states that *only publications may have authors*, and the triple (hasAuthor, rdfs:range, Researcher) states that *only researchers may be authors of something*.

Notations. For conciseness, we use the following shorthands for RDFS 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 a $\mathtt{p}$-labeled directed edge from the $\mathtt{s}$ node to the $\mathtt{o}$ node: $\mathtt{s} \xrightarrow{\mathtt{p}} \mathtt{o}$. 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, it is worth noticing the deductive nature of ontological constraints, which begets implicit triples within an RDF graph. For instance, in Fig. 1, the constraint $(\text{hasContactAuthor}, \prec_{\text{sp}}, \text{hasAuthor})$ together with the triple $(b, \text{hasContactAuthor}, b_1)$ imply the implicit triple $(b, \text{hasAuthor}, b_1)$, which, further, with the constraint $(\text{hasAuthor}, \hookrightarrow_r, \text{Researcher})$ yields another implicit triple $(b_1, \tau, \text{Researcher})$.

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 entire 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`).

The *saturation (or closure)* of an RDF graph $\mathcal{G}$ w.r.t. a set $\mathcal{R}$ of RDF entailment rules (a.k.a. entailment regime) 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 fixpoint reached by repeatedly applying the rules in $\mathcal{R}$ to $\mathcal{G}$ in a forward-chaining fashion, while adding to $\mathcal{G}$ the triples they derive. In RDF, the saturation is *finite*, *unique* (up to blank node renaming), and can be *computed in polynomial time* [25].

The saturation of the RDF graph $\mathcal{G}$ shown in Fig. 1 corresponds to the RDF graph $\mathcal{G}^\infty$ in which all the $\mathcal{G}$ implicit triples (dashed edges) are made explicit (solid edges). 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 above for deriving the triple $(b_1, \tau, \text{Researcher})$. This is automated by the following sequence of applications of RDF entailment rules: $(\text{hasContactAuthor}, \prec_{\text{sp}}, \text{hasAuthor})$ and $(b, \text{hasContactAuthor}, b_1)$ trigger `rdfs7` that adds $(b, \text{hasAuthor}, b_1)$ to the RDF graph. In turn, this new triple together with $(\text{hasAuthor}, \hookrightarrow_r, \text{Researcher})$ triggers `rdfs3` that adds $(b_1, \tau, \text{Researcher})$.

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 $\texttt{Bl}(\mathcal{G}')$ to $\texttt{Val}(\mathcal{G}^\infty)$ 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 $\mathcal{G}'$ entailed by the RDF graph $\mathcal{G}$ in Fig. 1 w.r.t. the 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 SPARQL". 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 at hand).

Importantly, from the definition of entailment between two RDF graphs [24,25], the following holds:

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

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

2.2 SPARQL Conjunctive Queries

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 select-project-join queries for databases; it is the most widely used subset of SPARQL queries in real-world applications [19].

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 *head* of q, noted $head(q)$, is $q(\bar{x})$, and the *body* of q, noted $body(q)$, is the BGP $\{t_1, \ldots, t_\alpha\}$ the cardinality of which is the *size* of q. 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 cardinality of $\bar{x}$ is the *arity* of q. We use x and y in queries, possibly with subscripts, for answer and non-answer variables respectively. Finally, we note $\texttt{VarBl}(q)$ the set of variables *and* blank nodes occurring in the query q, and $\texttt{Val}(q)$ the set of all its values, i.e., URIs, blank nodes, literals and variables.

Entailing and Answering Queries. Two related important notions characterize how an RDF graph contributes to a query.

The weaker notion, called *query entailment*, 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 an RDF 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, noted $\mathcal{G} \models_{\mathcal{R}} q$, iff $\mathcal{G} \models_{\mathcal{R}} body(q)$ holds, i.e., there exists a homomorphism ϕ from $\texttt{VarBl}(q)$ to $\texttt{Val}(\mathcal{G}^\infty)$ such that $[body(q)]_\phi \subseteq \mathcal{G}^\infty$.

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

Notations. Similarly to entailment between RDF graphs, 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.

The stronger notion characterizing how an RDF graph contributes to a query, called *query answering*, identifies *all* the query answers that this graph holds. Formally, given a BGPQ q with set $\bar{x}$ of answer variables, 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 $(\bar{x})_\phi$ is the tuple of $\mathcal{G}^\infty$ values obtained by replacing every answer variable $x_i \in \bar{x}$ by its image $\phi(x_i)$. In case of a Boolean query, q is false iff $q(\mathcal{G}) = \emptyset$; otherwise q is true and $q(\mathcal{G}) = \{\langle\rangle\}$ where $\langle\rangle$ denotes the empty tuple.

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 Fig. 1 is:

- $\{\langle b, \text{ConfPaper}\rangle, \langle b, \text{Publication}\rangle, \langle b_1, \text{Researcher}\rangle\}$ for $\mathcal{R}$ the set of entailment rules in Table 2, i.e., considering the explicit *and* implicit triples in Fig. 1;
- $\{\langle b, \text{ConfPaper}\rangle\}$ for $\mathcal{R} = \emptyset$, i.e., considering only the explicit triples in Fig. 1.

Importantly, from the definition of answer set of a SPARQL query against an RDF graph [26], the following holds:

Property 2. Given an RDF graph $\mathcal{G}$, a set $\mathcal{R}$ of entailment rules and a BGPQ q, (i) $\mathcal{G} \models_{\mathcal{R}} q$ holds iff $\mathcal{G}^{\infty} \models q$ holds, and (ii) $q(\mathcal{G}) = q(\mathcal{G}^{\infty})$ holds.

From a practical viewpoint, Property 2 points out that query entailment $\mathcal{G} \models_{\mathcal{R}} q$, respectively query answering $q(\mathcal{G})$, can be done in two steps: a reasoning step that computes the saturation $\mathcal{G}^{\infty}$ of $\mathcal{G}$, followed by a standard graph homomorphism step that checks if $\mathcal{G}^{\infty} \models^{\phi} q$ holds for some homomorphism ϕ, respectively enumerates all the homomorphisms ϕ for which $\mathcal{G}^{\infty} \models^{\phi} q$ holds.

Comparing Queries. Similarly to RDF graphs, queries can be compared through the generalization/specialization relationship of *entailment between queries*.

Let q, q' be BGPQs with the *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 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 $\texttt{VarBl}(body(q'))$ to $\texttt{Val}(body(q)^{\infty})$ such that $[body(q')]_{\phi} \subseteq body(q)^{\infty}$; the saturation of a BGP body, here $body(q)^{\infty}$, is the obvious generalization of RDF graph saturation that treats variables as blank nodes, since they both equivalently model unknown information within BGPs [26].

For instance, the query $q_1(x) \leftarrow (x, \tau, \text{ConfPaper}), (x, \text{hasContactAuthor}, y)$ entails the query $q_2(x) \leftarrow (x, \tau, y)$ with $\phi(x) = x$, $\phi(y) = \text{ConfPaper}$ and any set of entailment rules.

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

Property 3. Given an RDF graph $\mathcal{G}$, a set $\mathcal{R}$ of entailment rules and two BGPQs q, q' such that $q \models_{\mathcal{R}} q'$, (i) if $\mathcal{G} \models_{\mathcal{R}} q$ holds then $\mathcal{G} \models_{\mathcal{R}} q'$ holds, and (ii) $q(\mathcal{G}) \subseteq q'(\mathcal{G})$ holds.

Finally, query entailment, query answering and entailment between queries *treat blank nodes in queries exactly as non-answer variables* [26]. Hence, hereafter, we assume *without loss of generality* that queries do not use blank nodes.

3 Problem Statement

A *least general generalization* (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 [21]. In our SPARQL setting, we may use off-the-shelf BGPQs as descriptions and entailment between BGPQs as generalization/specialization relation:

Definition 1 (lgg of BGPQs). *Let $q_1, \ldots, q_n$ be BGPQs with the same arity and $\mathcal{R}$ a set of RDF entailment rules.*

– *A* generalization *of* $q_1, \ldots, q_n$ *is a BGPQ* q_g *such that* $q_i \models_{\mathcal{R}} q_g$ *for* $1 \leq i \leq n$.
– *A* least general generalization *of* $q_1, \ldots, q_n$ *is a generalization* $q_{\texttt{lgg}}$ *of* $q_1, \ldots, q_n$ *such that for any other generalization* q_g *of* $q_1, \ldots, q_n$*:* $q_{\texttt{lgg}} \models_{\mathcal{R}} q_g$.

Unfortunately, this straightforward definition is of limited practical interest as the next example shows. Consider the BGPQs q_1 and q_2 in Fig. 3, which respectively ask for *the conference papers having some contact author*, and for *the journal papers having some author*. Clearly, with the RDF entailment rules shown in Table 2, an `lgg` of q_1 and q_2 is the *very* general BGPQ $q_{\texttt{lgg}}(x) \leftarrow (x, \tau, y)$ asking for *the resources having some type*.

We argue that the value of `lgg`s could be significantly augmented by taking into account some *background knowledge* formalized as ontological constraints. For example, if we consider the RDFS statements shown in Fig. 3 that hold in the scientific publication domain, a more precise `lgg` for the above-mentioned q_1, q_2 would be $q_{\texttt{lgg}}(x) \leftarrow (x, \tau, \text{Publication}), (x, \text{hasAuthor}, y), (y, \tau, \text{Researcher})$ asking for *the publications having some researcher as author*, since (i) having a contact author is having an author, (ii) only publications have authors, (iii) only researchers are authors, and (iv) conference and journal papers are publications.

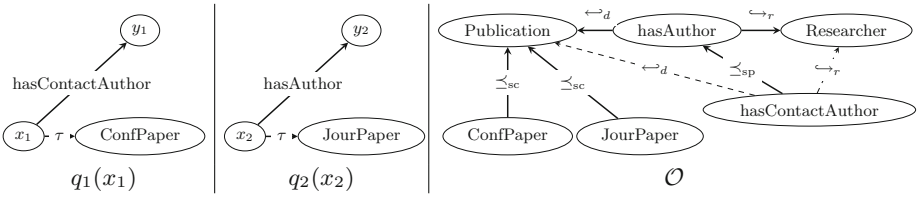

Fig. 3. Sample BGPQs q_1 and q_2; sample set $\mathcal{O}$ of RDFS ontological constraints.

To define such more precise `lgg`s and state our learning problem in Sect. 3.2, we start by generalizing the standard specialization/generalization relation of *entailment between BGPQs* in Sect. 3.1, in order to allow comparing BGPQs w.r.t. an extra set of RDFS ontological constraints. In particular, this novel relation (i) *coincides with the standard one* when extra constraints are unavailable and (ii) *behaves like the standard one* w.r.t. the central reasoning tasks of query entailment and of query answering when extra constraints are available.

3.1 Comparing Queries w.r.t. Ontological Constraints

Our new entailment relation between queries builds on the following notion, which leverages the relevant background knowledge to complement a query:

Definition 2 (BGPQ saturation w.r.t. RDFS constraints). *Let* $\mathcal{R}$ *be a set of RDF entailment rules,* $\mathcal{O}$ *a set of RDFS statements, and* q *a BGPQ. The* saturation *of* q *w.r.t.* $\mathcal{O}$*, noted* $q^{\infty}_{\mathcal{O}}$*, is the BGPQ with the same answer variables as* q *and whose body, noted* $body(q^{\infty}_{\mathcal{O}})$*, is the maximal subset of* $(body(q) \cup \mathcal{O})^{\infty}$ *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.*

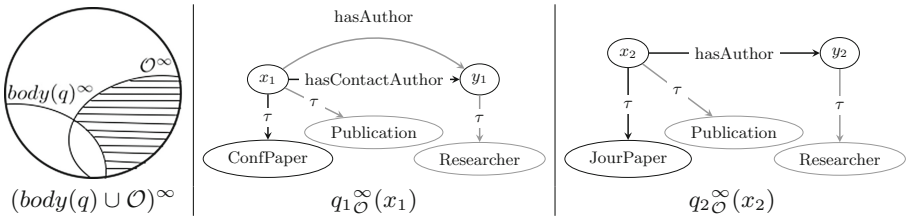

Fig. 4. Characterization of the body of a saturated BGPQ q w.r.t. a set $\mathcal{O}$ of RDFS constraints (left), and saturations of q_1 and of q_2 w.r.t. $\mathcal{O}$ from Fig. 3 (center and right respectively); triples shown in gray are added by saturation.

In essence, the saturation of a BGPQ comprises all the triples in the saturation of its body together with the RDFS constraints, from which are pruned out the triples derived solely from the constraints, i.e., which are not related to what the query is asking for. This corresponds exactly to the *non-hatched* subset of $(body(q) \cup \mathcal{O})^\infty$ shown in Fig. 4: $body(q^\infty_\mathcal{O}) = (body(q) \cup \mathcal{O})^\infty \setminus (\mathcal{O}^\infty \setminus body(q)^\infty)$. Of course, such a saturation is pertinent just in case the RDF entailment rules under consideration utilize the RDFS constraints, e.g., those in Table 2; otherwise the set of constraints is useless.

Figure 4 illustrates the saturation of queries w.r.t. ontological constraints using the BGPQs and RDFS contraints from Fig. 3.

The next theorem states that a BGPQ and its saturation w.r.t. RDFS constraints are *equivalent from the query entailment and query answering viewpoints*:

Theorem 1. *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 $q^\infty_\mathcal{O}$. For any RDF graph $\mathcal{G}$ whose set of RDFS statements is $\mathcal{O}$, (i) $\mathcal{G} \models_\mathcal{R} q$ holds iff $\mathcal{G} \models_\mathcal{R} q^\infty_\mathcal{O}$ holds, and (ii) $q(\mathcal{G}) = q^\infty_\mathcal{O}(\mathcal{G})$ holds.*

We can now endow entailment between queries with background knowledge:

Definition 3 (Entailment between BGPQs w.r.t RDFS 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 $q^\infty_\mathcal{O} \models q'$ holds.*

Using the set $\mathcal{R}$ of entailment rules in Table 2, the above mentioned BGPQ $q_{1\mathtt{gg}}(x) \leftarrow (x, \tau, \text{Publication}), (x, \text{hasAuthor}, y), (y, \tau, \text{Researcher})$ is *neither* entailed by q_1 *nor* by q_2 from Fig. 3, while it *is* entailed by both of them w.r.t. the set $\mathcal{O}$ of constraints displayed in the same Figure, i.e., it is entailed *in the standard fashion* by their saturations shown in Fig. 4: $q_1{}^\infty_\mathcal{O} \models^{\phi_1} q$ holds for $\phi_1(x) = x_1$ and $\phi_1(y) = y_1$, and $q_2{}^\infty_\mathcal{O} \models^{\phi_2} q$ holds for $\phi_2(x) = x_2$ and $\phi_2(y) = y_2$.

Clearly, the above definition *coincides* with standard RDF entailment between BGPQs when $\mathcal{O}$ is empty (recall Sect. 2). Further, the main theorem

below states the *required behaviour* for a query entailed by another w.r.t. ontological constraints, i.e., the counterpart of Property 3 in Sect. 2: *the former generalizes the latter from the query entailment and query answering viewpoints*:

Theorem 2. *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 $\mathcal{O}$, (i) if $\mathcal{G} \models_{\mathcal{R}} q$ holds then $\mathcal{G} \models_{\mathcal{R}} q'$ holds, and (ii) $q(\mathcal{G}) \subseteq q'(\mathcal{G})$ holds.*

3.2 Learning lggs w.r.t. Ontological Contraints

In the light of the preceding results, we revise/generalize Definition 1 as follows:

Definition 4 (lgg of BGPQs w.r.t RDFS constraints). *Let $\mathcal{R}$ be a set of RDF entailment rules, $\mathcal{O}$ a set of RDFS statements, and $q_1, \ldots, q_n$ BGPQs with the same arity.*

- *A* generalization of $q_1, \ldots, q_n$ *w.r.t. $\mathcal{O}$ is a BGPQ q_g such that $q_i \models_{\mathcal{R},\mathcal{O}} q_g$ 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. $\mathcal{O}$: $q_{\texttt{lgg}} \models_{\mathcal{R},\mathcal{O}} q_g$.*

By constrast with an lgg of RDF graphs that always exists [9], we found:

Theorem 3. *An* lgg *of BGPQs w.r.t. RDFS statements may not exist for some set of RDF entailment rules; when it exists, it is unique up to entailment ($\models_{\mathcal{R},\mathcal{O}}$).*

Indeed, consider the BGPQs $q_1(x_1) \leftarrow (x_1, \text{hasAuthor}, y_1)$ asking for the resources having some author, and $q_2(x_2) \leftarrow (y_2, \text{hasAuthor}, x_2)$ asking for the authors of some resource. Clearly, when the set $\mathcal{R}$ of entailment rules is empty or comprises the rules in Table 2, *no* BGPQ can generalize q_1 and q_2, hence there is *no* lgg of them. By contrast, if we use the complete set of RDF entailment rules, an lgg of q_1 and q_2 is $q_{\texttt{lgg}}(x) \leftarrow (x, \tau, \text{rdf:Resource})$, since every RDF value is an instance of the built-in class rdf:Resource. Also, when an lgg of BGPQs w.r.t. RDFS constraints exists, it is unique up to entailment, i.e., is semantically unique, because $q_{\texttt{lgg}} \models_{\mathcal{R},\mathcal{O}} q_g$ holds for any q_g in Definition 1. If it were that queries have *multiples* lggs *incomparable* w.r.t. entailment, say the BGPQs $\texttt{lgg}_1(\bar{x}), \ldots, \texttt{lgg}_m(\bar{x})$, the BGPQ defined as $q_{\texttt{lgg}}(\bar{x}) \leftarrow \mathit{body}(\texttt{lgg}_1) \cup \cdots \cup \mathit{body}(\texttt{lgg}_m)$ would be a *single strictly more specific* lgg, a contradiction.

Though unique up to entailment, there exist many syntactic variants (an infinity actually) of an lgg due to *redundant* triples, i.e., triples entailed by others within the lgg. For example, think of an lgg $q_{\texttt{lgg}}(x) \leftarrow (x, \tau, A), (x, \tau, B), (x, y, z)$ w.r.t. the set of constraints $\mathcal{O} = \{(A, \preceq_{\text{sc}}, B), (B, \preceq_{\text{sc}}, A)\}$, which asks for resources of types A and B that are somehow related to some resource, and it is known that A and B are equivalent classes. Clearly, different equivalent and minimal variants (w.r.t. the number of triples) of this lgg are $q_{\texttt{lgg}}(x) \leftarrow (x, \tau, A)$ and $q_{\texttt{lgg}}(x) \leftarrow$

(x, τ, B), since (x, y, z) is entailed by each of the two other triples, and (x, τ, B) is entailed by (x, τ, A) w.r.t. $\mathcal{O}$, and vice versa, because A and B are equivalent. Importantly, redundancy of triples is not specific to `lgg`s of BGPQs w.r.t. RDFS constraints, since obviously any BGPQ may feature redundancy. The detection and elimination of such redundancy have been studied in the literature [18,20], hence we focus in this work on learning *some* `lgg` of BGPQs w.r.t. RDFS constraints; learning as minimal as possible `lgg`s is a perspective of this work discussed in Sect. 6.

Based on the above discussion, the learning problem we propose to study is:

Problem 1. Given a set $\mathcal{R}$ of RDF entailment rules, a set $\mathcal{O}$ of RDFS statements, and the BGPQs $q_1, \ldots, q_n$ with the same arity, find an `lgg` of $q_1, \ldots, q_n$ w.r.t. $\mathcal{O}$.

Importantly, the proposition below shows that an `lgg` of $n \geq 3$ BGPQs can be inductively defined, hence computed, as a sequence of $n-1$ `lgg`s 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:

$$\begin{aligned}
\text{[basis]} \quad \ell_3(q_1, q_2, q_3) &\equiv_{\mathcal{R},\mathcal{O}} \ell_2(\ell_2(q_1, q_2), q_3) \\
\text{[induction]} \quad \ell_n(q_1, \ldots, q_n) &\equiv_{\mathcal{R},\mathcal{O}} \ell_2(\ell_{n-1}(q_1, \ldots, q_{n-1}), q_n) \\
&\equiv_{\mathcal{R},\mathcal{O}} \ell_2(\ell_2(\cdots \ell_2(\ell_2(q_1, q_2), q_3) \cdots, q_{n-1}), q_n)
\end{aligned}$$

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

Based on the above result, *without loss of generality*, we study in the next Section the particular instance of our learning problem for $n = 2$.

4 Computing `lgg`s of Queries w.r.t. Ontological Constraints

Our solution to the above learning problem (Problem 1) builds on the notion of *least general anti-unifier of two atoms* [21,23], which is dual to the well-known notion of *most general unifier of two atoms* [22,23]. We use it to devise the *cover query* of two BGPQs q_1 and q_2 (to be defined shortly, Definition 5 below), which is an `lgg` of q_1 and q_2 *just in case* both RDF entailment rules and ontological constraints are ignored (Theorem 4). Further, we show (Theorem 5) that an `lgg` of q_1 and q_2 as defined in Definition 4, i.e., when RDF entailment rules and ontological constraints are taken into consideration, is the cover query of the saturations of q_1 and of q_2 with the RDF entailment rules and ontological constraints at hand (Definition 2). We also provide the size of these cover query-based `lgg`s (i.e., number of triples), as well as the time to compute them.

Definition 5 (Cover query). *Let q_1, q_2 be two BGPQs with the same arity n. If there exists the BGPQ q such that*

- *$head(q_1) = q(x_1^1, \ldots, x_1^n)$ and $head(q_2) = q(x_2^1, \ldots, x_2^n)$ iff $head(q) = q(v_{x_1^1 x_2^1}, \ldots, v_{x_1^n x_2^n})$*

– $(t_1, t_2, t_3) \in body(q_1)$ *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* $v_{t_i t_{i+3}}$

then q *is the* cover query *of* q_1, q_2.

The rationale behind the above definition of cover query is that (*i*) q's head is defined as the *least general anti-unifier* of the heads of q_1 and q_2 (first item above) and (*ii*) each q triple is defined as a *least general anti-unifier* of an *explicit* q_1 triple and an *explicit* q_2 triple (second item above), so that, when the cover query exits (If ... then ... above), it is a *generalization* of q_1 and q_2 just in case RDF entailment rules and ontological constraints are *not considered* (first item in Definition 4 with $\mathcal{R} = \emptyset$ and $\mathcal{O} = \emptyset$). Moreover, crucially, (*iii*) the variables used to generalize pairs of distinct values *across all* the anti-unifications begetting q are *consistently named*: each time the distinct values α from q_1 and β from q_2 are generalized by a variable across these anti-unifications, it is *always* by the same q variable $v_{\alpha\beta}$. This naming scheme enforces *joins* between q triples, which capture the common join structure within q_1 and q_2, so that q is not only a generalization of q_1 and q_2 but also a *least general generalization* of them (second item in Definition 4 with $\mathcal{R} = \emptyset$ and $\mathcal{O} = \emptyset$).

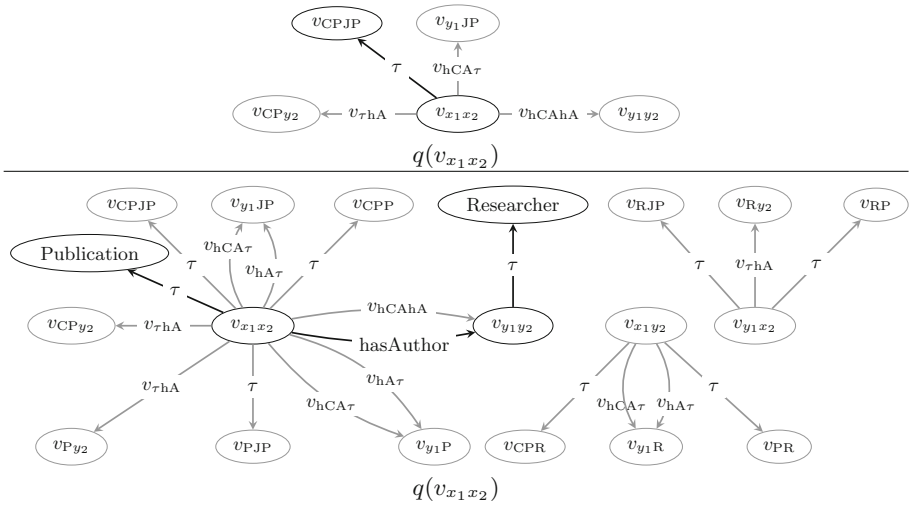

Fig. 5. Cover queries of the BGPQs q_1 and q_2 in Fig. 3 (top) and of their saturations $q_1^{\infty}_{\mathcal{O}}$ and $q_2^{\infty}_{\mathcal{O}}$ in Fig. 4 (bottom). Triples in grey are redundant w.r.t. those in black.

The cover query q of the BGPQs q_1 and q_2 is displayed in Fig. 5 (top). Its triple $(v_{x_1x_2}, \tau, v_{\text{CPJP}})$ results from anti-unifying the q_1 triple $(x_1, \tau, \text{ConfPaper})$ and the q_2 triple $(x_2, \tau, \text{JourPaper})$; the variable $v_{x_1x_2}$ is the least general value for the subject values x_1 and x_2, the URI τ is that for the property values τ

(because a constant is the least generalization of itself), and the variable v_{CPJP} is that for the object values ConfPaper and JourPaper. This q triple captures that q_1 and q_2 both ask for resources having some type. Here, the fact this type is related to scientific publications is missed, due to the absence of background knowledge relating conference papers, journal papers and scientific publications. Similarly, the q triple $(v_{x_1x_2}, v_{\text{hCAhCA}}, v_{y_1y_2})$ results from anti-unifying the q_1 triple $(x_1, \text{hasContactAuthor}, y_1)$ and the q_2 triple $(x_2, \text{hasAuthor}, y_2)$. Because of our consistent naming of variables within q, this q triple and the preceding one join on $v_{x_1x_2}$. Unfortunately, this second triple does not enhance the description of $v_{x_1x_2}$ in q, since it generalizes, hence is redundant with, the preceding one. It only captures from q_1 and q_2 that q asks for resources having somehow related to something. Here again, the fact that this relationship is to have some author is missed due to the absence of background knowledge. The two other anti-unifications begetting q's body also produce redundant triples.

As mentioned earlier, the cover query q of two BGPQs q_1 and q_2 may not exist. This happens when q, as defined in Definition 5, has its head *not compatible* with its body: some required answer variable(s) cannot be supplied by q's body. For instance, recall the BGPQs q_1 and q_2 used in Sect. 3.2 to point out that an `lgg` may no exist. Their cover query does not exist either, because Definition 5 leads to $q(v_{x_1x_2}) \leftarrow (v_{x_1y_2}, \text{hasAuthor}, v_{y_1x_2})$, which is *not* a BGPQ (the answer variable $v_{x_1x_2}$ does not appear in the body). Importantly, the existence of an `lgg` of BGPQs and the existence of their cover query *coincide*.

The next theorem formalizes the above discussion:

Theorem 4. *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.*

It follows from the above result that the cover query q of two BGPQs q_1 and q_2 displayed in Fig. 5 (top) is an `lgg` of them *just in case* both RDF entailment rules and extra RDFS ontological constraints are ignored.

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

Proposition 2. *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 RDFS ontological constraints* and *any* set of RDF entailment rules.

Theorem 5. *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 ${q_1}^{\infty}_{\mathcal{O}}, {q_2}^{\infty}_{\mathcal{O}}$ exists iff an* `lgg` *of q_1, q_2 w.r.t. $\mathcal{O}$ exists;*
2. *the cover query q of ${q_1}^{\infty}_{\mathcal{O}}, {q_2}^{\infty}_{\mathcal{O}}$ is an* `lgg` *of q_1, 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 RDFS 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 sets $\mathcal{O}$ and $\mathcal{R}$ at hand.

Corollary 1. *A cover query-based* lgg *of two BGPQs* q_1 *and* q_2 *is 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 5 (bottom) displays the cover query of the BGPQs $q_1^{\infty_{\mathcal{O}}}$ and $q_2^{\infty_{\mathcal{O}}}$ shown in Fig. 4. It is therefore (Theorem 5) an lgg of the BGPQs q_1 and q_2 w.r.t. the set $\mathcal{O}$ of RDFS constraints, all shown in Fig. 3, using the RDF entailment rules shown in Table 2.

Figure 5 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 (top), 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 (bottom), 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

Goal. We study the added-value of considering background knowledge when learning lggs of queries. As Proposition 3 shows below, this amounts to measuring *how much more precise* is an lgg of queries that considers background knowledge *than* an lgg of the same queries that ignores background knowledge:

Proposition 3. *Given a set* $\mathcal{R}$ *of RDF entailment rules, a set* $\mathcal{O}$ *of RDFS statements, two BGPQs* q_1, q_2 *with the same arity, an* lgg $q_{\texttt{lgg}}$ *of* q_1, q_2 *(Definition 1) and an* lgg $q_{\texttt{lgg}}^{\mathcal{O}}$ *of* q_1, q_2 *w.r.t.* $\mathcal{O}$ *(Definition 4),* $q_{\texttt{lgg}}^{\mathcal{O}} \models_{\mathcal{R}} q_{\texttt{lgg}}$ *holds.*

Intuitively, this result follows from the fact that (*i*) $q_{\texttt{lgg}}$ is *equivalent* to the cover query-based lgg q of the saturations of q_1 and of q_2 w.r.t. the empty set of RDFS constraints, (*ii*) $q_{\texttt{lgg}}^{\mathcal{O}}$ is *equivalent* to the cover query-based lgg q' of the saturations of q_1 and of q_2 w.r.t. $\mathcal{O}$, and (*iii*) *by definition* of a cover query (Definition 5), q and q' have the same heads and the body of q is a subset of that q', thus $q' \models_{\mathcal{R}} q$ holds, hence $q_{\texttt{lgg}}^{\mathcal{O}} \models_{\mathcal{R}} q_{\texttt{lgg}}$ holds.

From this result and Property 3 (Sect. 2.2), $q_{\texttt{lgg}}^{\mathcal{O}}(\mathcal{G}) \subseteq q_{\texttt{lgg}}(\mathcal{G})$ holds for any RDF graph $\mathcal{G}$, and clearly the more $q_{\texttt{lgg}}^{\mathcal{O}}$ is specific w.r.t. $q_{\texttt{lgg}}$, the smaller the subset $q_{\texttt{lgg}}^{\mathcal{O}}(\mathcal{G})$ of $q_{\texttt{lgg}}(\mathcal{G})$ is, i.e., the smaller $|q_{\texttt{lgg}}^{\mathcal{O}}(\mathcal{G})|$ is w.r.t. $|q_{\texttt{lgg}}(\mathcal{G})|$. Therefore, as a practical metric for measuring the semantic distance between $q_{\texttt{lgg}}^{\mathcal{O}}$ and $q_{\texttt{lgg}}$ through $\models_{\mathcal{R}}$, we compute the *gain in precision* in (%) that background knowledge yields *w.r.t. query answering* as:

$$\text{gain in precision} = 1 - \frac{|q^{\mathcal{O}}_{\mathtt{1gg}}(\mathcal{G}) \cap q_{\mathtt{1gg}}(\mathcal{G})|}{|q_{\mathtt{1gg}}(\mathcal{G})|} = 1 - \frac{|q^{\mathcal{O}}_{\mathtt{1gg}}(\mathcal{G})|}{|q_{\mathtt{1gg}}(\mathcal{G})|} \text{ since } q^{\mathcal{O}}_{\mathtt{1gg}}(\mathcal{G}) \subseteq q_{\mathtt{1gg}}(\mathcal{G}).$$

Prototype. We implemented our technical contributions in Java 1.8, on top of the Jena 3.0.1 RDF reasoner and of a PostgreSQL 9.3.11 server, all used with default settings; our implemented algorithms are detailed in [8].

We used Jena to compute the *saturation of an RDF graph*, against which queries must be evaluated to obtained their *complete* answer sets (Sect. 2.1). We also used Jena to compute the *saturation* $q^{\infty}_{\mathcal{O}}$ *of a BGPQ* q w.r.t. *a set* $\mathcal{O}$ *of RDFS constraints* (Definition 2): we rely on Jena's saturation, union and difference operators to compute $q^{\infty}_{\mathcal{O}}$'s body as described in Sect. 3.1.

We used PostgreSQL to evaluate *SQLized* BGPQs against a *saturated* RDF graph stored in a `Triple(s,p,o)` table.

We deployed our prototype on an Intel Xeon X5550 2.67 GHz machine with 32 GB of RAM, running Ubuntu 14.04.3 64bits; times reported below are in ms.

Setting. We conducted experiments using *real DBpedia data* [17] and *synthetic LUBM data* [12]. For space reasons, we present only our DBpedia experiments; LUBM ones can be found in [8] and allow drawing similar conclusions.

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

From the DBpedia dataset, we picked four complementary files[1] to build the RDF graph $\mathcal{G}_{\mathtt{DBpedia}}$ comprising 41.18M triples, whose subset $\mathcal{O}_{\mathtt{DBpedia}}$ of 30.31k RDFS constraints represents DBpedia's background knowledge. The saturation of $\mathcal{G}_{\mathtt{DBpedia}}$ comprises 78.14M triples and takes about 30 min to be computed.

Finally, we defined 42 test BGPQs, among which we picked 8 representative ones with 2 variables; they can be found in [8]. Table 3 displays their characteristics (top), as well as their saturation size and time (bottom): the size augments from $\times 3.16$ for Q_2 up to $\times 4.75$ for Q_3; the time is 692 ms on average. Also, importantly, queries Q_1–Q_4 (left) are heterogeneous in the sense that they differ significantly both on their structure and the kind of information they ask for, hence use many distinct classes, properties and URI values, while Q_4–Q_8 (right) are homogeneous and only differ in some classes, properties and URI values.

Table 3. Characteristics of our test BGPQs (top) and of their saturations (bottom).

Query $Q_{1 \leq i \leq 8}$:	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8
Q_i's shape	tree	tree	tree	graph	graph	graph	graph	graph
$\|body(Q_i)\|$	4	6	4	6	4	6	6	6
Number of URI/variable occurrence in Q_i	7/5	9/9	5/7	7/11	5/7	9/9	9/9	9/9
$\|Q_i(\mathcal{G}_{\mathtt{DBpedia}})\|$	77	0	41,695	13	6	0	1	0
$\|body(Q_i{}^{\infty}_{\mathcal{O}_{\mathtt{DBpedia}}})\|$	16	19	19	23	16	23	23	23
Time to compute $Q_i{}^{\infty}_{\mathcal{O}_{\mathtt{DBpedia}}}$	666	643	677	734	681	706	697	736

[1] We use the `dbpedia_2015-10.nt` RDF Schema file and the `instance_types_en.ttl`, `mappingbased_literals_en.ttl` and `mappingbased_objects_en.ttl` RDF data files.

Table 4. Characteristics of cover query-based `lgg`s of test queries, w/ or w/o using the DBpedia RDFS constraints.

`lgg` of:	Q_1Q_2	Q_1Q_3	Q_1Q_4	Q_2Q_3	Q_4Q_5	Q_5Q_6	Q_5Q_7	Q_7Q_8
Time to compute $q_{\mathtt{lgg}}$	3	3	5	4	4	5	6	5
$\lvert q_{\mathtt{lgg}}(\mathcal{G}_{\mathtt{DBpedia}})\rvert$	477,455	34,747,102	34,901,117	60,356,807	1,977	1,221	35	70
Time to compute $q_{\mathtt{lgg}}^{\mathcal{O}_{\mathtt{DBpedia}}}$	13	14	14	15	15	14	17	18
$\lvert q_{\mathtt{lgg}}^{\mathcal{O}_{\mathtt{DBpedia}}}(\mathcal{G}_{\mathtt{DBpedia}})\rvert$	10,637	7,874,768	456,690	7,874,768	1,701	780	34	36
Gain in precision	98	77	99	87	14	36	3	49

Results. First, as Table 4 (lines 1 and 3) shows, the cover query-based `lgg`s of test queries are always computed fast whether or not the DBpedia constraints are considered: from 3 to 6 ms when ignored, to 13 to 18 ms when considered.

Table 4 (lines 2 and 4) also shows that the answer set of an `lgg` is significantly larger when DBpedia constraints are not taken into account: the size difference goes from a small $\times 1.02$ for the homogeneous queries Q_5, Q_7 up to a striking $\times 76.42$ for the heterogeneous queries Q_1, Q_4, with a significant average of $\times 17.38$ ($\times 33.34$ for the heterogeneous queries and $\times 1.42$ for the homogeneous ones). This translates into the precision gains shown at line 5: 58% overall, 90% for the heterogeneous queries, and 25% for the homogeneous ones.

These results confirm our claim that *taking into account background knowledge yields more precise* `lgg`*s*. Indeed, ontological constraints help finding *common super-* classes and properties to be used in `lgg`s in place of the different ones used in input queries; when constraints are ignored, these can just be generalized using *variables*. Therefore, the more heterogeneous input queries are, the more such common super-classes and properties are used in their `lgg` instead of variables, and the more the gain in precision of their `lgg` is high. For homogeneous input queries, while less striking, the gain in precision is significant in general.

6 Related Work and Conclusion

The reasoning problem of learning `lgg`s has been studied in various formalisms, e.g., Conceptual Graphs (CGs), Description Logics (DLs), RDF and SPARQL.

Most of the solutions exploit the (underlying) *structure* of the input descriptions, like *trees* for DL formulae (e.g., [1,14,27]) and for unary tree-shaped BGPQs [2,16], and *directed single-root graphs* for the RDF r-graphs of [7]. Roughly speaking, they all consist in a simultaneous traversal of the input descriptions, starting from their roots, while incrementally computing their `lgg`. In contrast, when the input descriptions do not have a particular (or imposed) structure, solutions need to blindly traverse them while still being able to compute their `lgg`. They rely on standard *categorial graph product* for the so-called *simple* (i.e., purely conjunctive) CGs [3], on *anti-unifications of triples* for *general* RDF graphs [9], and on *anti-unifications of query heads and of query body triples* for the *general* BGPQs considered in this paper. Further, while (some of) the above solutions take into account background knowledge in CGs, DLs,

and RDF, this is not the case for the state of the art in SPARQL [2,16]: unary tree-shaped BGPQs are solely compared based on standard graph homomorphism ($\models_\emptyset$).

Our results significantly advance the state of the art [2,16] by considering (*i*) *general BGPQs* and (*ii*) *background knowledge* to obtain more precise `lgg`s, as our experiments showed. Next, we plan studying heuristics that prune out as much as possible redundant triples, *while computing* `lgg`*s*. Indeed, as Fig. 5 shows, our cover query-based `lgg`s may contain redundant triples. This would allow having more compact `lgg`s, as well as reducing the a posteriori elimination effort of redundant triples using standard technique from the literature.

References

1. Baader, F., Sertkaya, B., Turhan, A.Y.: Computing the least common subsumer w.r.t. a background terminology. J. Appl. Logic **5**(3), 392–420 (2007)
2. Bühmann, L., Lehmann, J., Westphal, P.: DL-Learner - a framework for inductive learning on the Semantic Web. J. Web Semant. **39**, 15–24 (2016)
3. Chein, M., Mugnier, M.: Graph-Based Knowledge Representation - Computational Foundations of Conceptual Graphs. Springer, London (2009)
4. Chuang, S.L., Chien, L.F.: Towards automatic generation of query taxonomy: a hierarchical query clustering approach. In: ICDM (2002)
5. Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: AAAI (1992)
6. Colazzo, D., Goasdoué, F., Manolescu, I., Roatis, A.: RDF analytics: lenses over semantic graphs. In: WWW (2014)
7. Colucci, S., Donini, F.M., Giannini, S., Sciascio, E.D.: Defining and computing least common subsumers in RDF. J. Web Semant. **39**, 62–80 (2016)
8. El Hassad, S., Goasdoué, F., Jaudoin, H.: Learning commonalities in RDF and SPARQL (research report) (2016). https://hal.inria.fr/hal-01386237
9. El Hassad, S., Goasdoué, F., Jaudoin, H.: Learning commonalities in RDF. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10249, pp. 502–517. Springer, Cham (2017). doi:10.1007/978-3-319-58068-5_31
10. El Hassad, S., Goasdoué, F., Jaudoin, H.: Towards learning commonalities in SPARQL. In: d'Amato, C., et al. (eds.) ESWC 2017. LNCS, vol. 10587, pp. 278–295. Springer, Cham (2017)
11. Goasdoué, F., Karanasos, K., Leblay, J., Manolescu, I.: View selection in semantic web databases. PVLDB **5**(2), 97–108 (2011)
12. Guo, Y., Pan, Z., Heflin, J.: LUBM: a benchmark for OWL knowledge base systems. J. Web Semant. **3**(2–3), 158–182 (2005)
13. Huang, Z., Cautis, B., Cheng, R., Zheng, Y.: KB-enabled query recommendation for long-tail queries. In: CIKM (2016)
14. Küsters, R.: Non-Standard Inferences in Description Logics. Lecture Notes in Computer Science, vol. 2100. Springer, Heidelberg (2001)
15. Le, W., Kementsietsidis, A., Duan, S., Li, F.: Scalable multi-query optimization for SPARQL. In: ICDE (2012)
16. Lehmann, J., Bühmann, L.: AutoSPARQL: let users query your knowledge base. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., Leenheer, P., Pan, J. (eds.) ESWC 2011. LNCS, vol. 6643, pp. 63–79. Springer, Heidelberg (2011). doi:10.1007/978-3-642-21034-1_5

17. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia. Semant. Web **6**(2), 167–195 (2015)
18. Meier, M.: Towards rule-based minimization of RDF graphs under constraints. In: Calvanese, D., Lausen, G. (eds.) RR 2008. LNCS, vol. 5341, pp. 89–103. Springer, Heidelberg (2008). doi:10.1007/978-3-540-88737-9_8
19. Picalausa, F., Luo, Y., Fletcher, G.H.L., Hidders, J., Vansummeren, S.: A structural approach to indexing triples. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 406–421. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30284-8_34
20. Pichler, R., Polleres, A., Skritek, S., Woltran, S.: Complexity of redundancy detection on RDF graphs in the presence of rules, constraints, and queries. Semant. Web **4**(4), 351–393 (2013)
21. Plotkin, G.D.: A note on inductive generalization. Mach. Intell. **5**, 153–163 (1970)
22. Robinson, J.A.: A machine-oriented logic based on the resolution principle. J. ACM **12**(1), 23–41 (1965)
23. Robinson, J.A., Voronkov, A. (eds.): Handbook of Automated Reasoning. Elsevier and MIT Press, Amsterdam and Cambridge (2001)
24. Resource description framework 1.1. https://www.w3.org/TR/rdf11-concepts
25. RDF 1.1 semantics. https://www.w3.org/TR/rdf11-mt/
26. SPARQL 1.1. https://www.w3.org/TR/sparql11-query/
27. Zarrieß, B., Turhan, A.: Most specific generalizations w.r.t. general EL-TBoxes. In: IJCAI (2013)