



HAL
open science

Learning commonalities in RDF & SPARQL

Sara El Hassad

► **To cite this version:**

Sara El Hassad. Learning commonalities in RDF & SPARQL. Programming Languages [cs.PL]. Université de Rennes, 2018. English. NNT : 2018REN1S011 . tel-01835703

HAL Id: tel-01835703

<https://theses.hal.science/tel-01835703v1>

Submitted on 11 Jul 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



**UNIVERSITE
BRETAGNE
LOIRE**

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de
DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale MathSTIC

présentée par

Sara EL HASSAD

préparée à l'unité de recherche IRISA – UMR6074
Institut de Recherche en Informatique et Système Aléatoires
École nationale supérieure des sciences appliquées et de technologie

Learning Commonalities in RDF & SPARQL

**Thèse soutenue à Lannion
le 2 Février 2018**

devant le jury composé de :

M FAROUK TOUMANI

PR, Université de Clermont-Ferrand / *Président*

Mme MARIE-CHRISTINE ROUSSET

PR, Université de Grenoble Alpes / *Rapporteur*

M JEAN-MARC PETIT

PR, INSA Lyon / *Rapporteur*

M FRANÇOIS GOASDOUÉ

PR, Université de Rennes 1 / *Directeur de thèse*

Mme HÉLÈNE JAUDOIN

MCF, Université de Rennes 1 / *Co-directeur de thèse*

A winner is a dreamer who never gives up
Nelson Mandela

Table des matières

Table of contents	0
Acknowledgements	5
Résumé en Français	7
Introduction	13
1 Preliminaries	17
1.1 Introduction	17
1.2 Resource Description Framework (RDF)	17
1.2.1 RDF graphs	17
1.2.2 Adding ontological knowledge to RDF graphs	18
1.2.3 Deriving the implicit triples of an RDF graph	19
1.2.4 Comparing RDF graphs.	21
1.3 SPARQL conjunctive queries	21
1.3.1 Basic graph pattern queries	21
1.3.2 Entailing and answering queries	22
1.3.3 Comparing queries	23
1.4 Conclusion	24
2 Finding Commonalities between RDF graphs	27
2.1 Introduction	27
2.2 Defining the lgg of RDF graphs	28
2.3 Computing an lgg of RDF graphs	31
2.4 Conclusion	35
3 Finding Commonalities between SPARQL queries	37
3.1 Introduction	37
3.2 Comparing Queries w.r.t. Ontological Constraints	39
3.3 Learning the lgg of Queries w.r.t. Ontological Constraints	42
3.4 Computing an lgg of Queries w.r.t. ontological constraints	44
3.5 Conclusion	49

4 Algorithms	51
4.1 Introduction	51
4.2 Least general anti-unification	51
4.3 lggs of RDF graphs	52
4.3.1 Handling large RDF graphs using DMSs	53
4.3.2 Handling huge RDF graphs using MapReduce	55
4.4 lggs of BGPQs	56
4.5 Conclusion	57
5 Experiments	59
5.1 Introduction	59
5.2 Gain in precision metrics	60
5.2.1 The case of RDF graphs	60
5.2.2 The case of BGPQs	61
5.3 Experimental setting	62
5.3.1 Software	62
5.3.2 Hardware	62
5.3.3 Datasets	62
5.4 lggs of RDF graphs	65
5.5 lggs of BGPQs	66
5.6 Conclusion	69
6 Related Work	71
6.1 Introduction	71
6.2 Inductive Logic Programming	71
6.3 Knowledge Representation	72
6.3.1 Description logics	72
6.3.2 Conceptual graphs	74
6.4 Semantic Web	74
6.4.1 RDF	74
6.4.2 SPARQL	75
6.5 Conclusion	76
7 Conclusion and Perspectives	79
7.1 Conclusion	79
7.2 Perspectives	80
7.2.1 Redundancy elimination	80
7.2.2 Learning commonalities in DL-Lite	80
Appendix	83
A DBpedia graphs	85
B LUBM queries	89

<i>Table des matières</i>	3
C DBpedia queries	91
D DBpedia extracted ontology \mathcal{O}	93
Bibliography	111
Table of figures	113
Table of tables	115
Liste of algorithms	117

Acknowledgements

First of all I would like to express my sincere gratitude to my thesis director François GOASDOUÉ and my co-supervisor Hélène JAUDOIN for all that they brought me on a professional and personal level, it was very pleasant to work in this team.

I also want to thank the president of the jury members Farouk TOUMANI, and I thank Marie Christine ROUSSET and Jean-Marc PETIT for agreeing to be the referees of this thesis.

I am thankful for all members of the Shaman team and the Expression team for their kindness, warm hospitality and encouragement, and I would like to thank all the people who have contributed from near or far to carrying out this research.

I would like to thank Bretagne region and Lannion agglomeration for funding this thesis.

Finally I thank my parents and my sister Yasmine for their enthusiasm and encouragement throughout my thesis, I especially thank my mother for making long trips to support me and attend my defense. I also thank my husband Anass for his presence, his enthusiasm and his support during my moments of doubt, and my daughter Lara who was the third contribution of this thesis and a real booster for my success.

Résumé en Français

Introduction

La recherche des points communs entre des descriptions de données ou des connaissances est un problème de raisonnement fondamental en Machine Learning qui a été formalisé par G. Plotkin dans les années 70s sous la forme du calcul du *plus petit généralisant* (*least general generalization*, noté **lgg**) de ces descriptions.

Dans les années 90, ce problème a également été étudié dans le domaine de la représentation des connaissances où le **lgg** a été rebaptisé *plus petit subsumer commun* (*least common subsumer*). Il a plus particulièrement été investigué dans les logiques des descriptions et les graphes conceptuels.

Récemment, ce problème a été exploré dans le domaine du web sémantique, notamment dans le modèle de données “graphe” Resource Description Framework (RDF) et dans son langage de requête associé SPARQL, qui sont deux standards du web sémantique du W3C.

Le **lgg** peut être utilisé dans de nombreuses applications importantes. Par exemple, en *optimisation de requêtes*, un **lgg** des requêtes reçues caractérise le plus grand ensemble de leurs points communs dont le traitement peut être factorisé. Dans le cadre d’un entrepôt des données, un **lgg** permet d’identifier les fragments de requêtes qui reviennent souvent et qui peuvent être matérialisés sous forme de vues. En *recommandation*, et plus particulièrement dans les réseaux sociaux, un **lgg** d’un ensemble des descriptions d’utilisateurs (profiles) peut aider à recommander les utilisateurs entre eux, à créer des groupes ou à former une communauté, si leurs centres d’intérêts sont assez proches. Par ailleurs, un **lgg** d’un ensemble de requêtes émises par les mêmes utilisateurs permet de les recommander entre eux si ce qu’ils cherchent est assez proche. En *classification*, un **lgg** d’un ensemble des données peut être utilisé pour les classer en considérant leurs informations communes. Il peut aussi être utilisé pour identifier la structure commune de certaines organisations (criminelles par exemple). Enfin, un **lgg** d’un ensemble de requêtes permet de les regrouper selon points communs et ainsi de recommander des recherches similaires ou complémentaires.

L’objectif de cette thèse est de revisiter le problème du calcul du **lgg** entre des descriptions du web sémantique, quand celles-ci sont des graphes RDF et des requêtes SPARQL, et pour lesquelles l’état de l’art ne fournit que des solutions structurellement

et sémantiquement limitées.

Préliminaires

Les contributions de cette thèse reposent sur RDF et le fragment conjonctif de SPARQL, *Basic Graph Pattern Queries (BGPQ)*.

Un graphe RDF est un ensemble de triplets de la forme (s, p, o) où s est le sujet qui a la propriété p et la valeur de cette propriété est l'objet o . Un triplet appartient à l'ensemble $(\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{L} \cup \mathcal{B})$, avec \mathcal{U} est l'ensemble des *identifiants uniformes de ressource* (URIs), \mathcal{B} est l'ensemble des ressources anonymes qui identifient les valeurs manquantes du graphe RDF et \mathcal{L} est l'ensemble des littéraux qui correspondent aux constantes dans un graphe RDF. Un triplet est une assertion de classe (relation unaire) ou de propriété (relation binaire). Un graphe RDF peut être enrichi avec des *contraintes ontologiques* entre les classes et leur propriétés. Une contrainte ontologique est un triplet qui contient une assertion *RDF Schema* (RDFS). Ces contraintes ontologiques permettent de dériver des triplets implicites. Ils sont rendus explicites en appliquant d'une manière exhaustive en chaînage avant sur un ensemble de règles d'implication \mathcal{R} fournies par le standard RDF. En ajoutant tous les triplets dérivés à notre graphe initial, nous obtenons un graphe stable et unique nommé la saturation du graphes RDF qui matérialise la sémantique du graphe RDF.

Le standard RDF définit une relation de généralisation/spécialisation entre graphes RDF nommée l'implication simple entre les graphes RDF (\models) qui permet de comparer les graphes en se basant sur leurs triplets explicites uniquement. Il a également défini une autre relation de généralisation/spécialisation entre graphes RDF nommée l'implication standard entre graphes RDF ($\models_{\mathcal{R}}$) qui permet de comparer les graphes selon leurs triplets explicites et implicites. Dans la suite, nous allons utiliser l'implication standard pour calculer les **lggs**.

Le langage de requête associé à RDF est SPARQL. Une requête SPARQL est composée de deux parties, la tête qui contient les variables de réponse et le corps qui est composé d'un ensemble de triplets. Un triplet du corps d'une requête SPARQL généralise un triplet du graphe RDF en autorisant les variables dans le sujet, la propriété et l'objet. Ainsi un triplet d'une requête SPARQL appartient à $(\mathcal{V} \cup \mathcal{U} \cup \mathcal{B}) \times (\mathcal{V} \cup \mathcal{U}) \times (\mathcal{V} \cup \mathcal{U} \cup \mathcal{L} \cup \mathcal{B})$ avec \mathcal{V} est l'ensemble des variables. Dans ce travail, nous considérons les ressources anonymes dans une requêtes SPARQL comme des variables.

Il existe deux notions importantes qui caractérisent comment un graphe RDF contribue à une requête : l'*implication d'une requête* q qui indique si oui ou non un graphe \mathcal{G} RDF fournit une réponse à la requête ($\mathcal{G} \models_{\mathcal{R}} q$) et *répondre à une requête* q qui identifie l'ensemble des réponses d'une requête dans un graphe ($q(\mathcal{G})$).

Comme pour les graphes RDF, il existe une relation de généralisation/spécialisation entre les requêtes SPARQL nommée l'implication standard entre les requêtes ($\models_{\mathcal{R}}$).

Trouver les points communs entre des graphes RDF

Le plus petit généralisant commun (**lgg**) entre des descriptions est la plus spécifique des descriptions qui généralise toutes les autres descriptions, selon une relation d'implication entre des descriptions.

Dans le contexte de RDF, les descriptions sont les graphes RDF et nous utilisons l'implication entre les graphes RDF comme relation de généralisation/spécialisation. Nous avons prouvé que le **lgg** de graphes existe toujours et qu'il est sémantiquement unique par rapport à l'implication.

Le calcul du **lgg** de n graphes est équivalent au calcul d'une séquence de $n-1$ **lgg** de deux graphes. Sans perte de généralité, nous avons étudié une technique pour calculer le **lgg** de deux graphes \mathcal{G}_1 et \mathcal{G}_2 . Pour cela, nous avons défini la notion de *graphe couvrant* qui est formé de l'ensemble des *anti-unifications* de chaque paire de triplets pris dans \mathcal{G}_1 et \mathcal{G}_2 respectivement. L'anti-unification désigne la transposition aux triplets RDF, de la notion d'anti-unification de deux atomes du premier ordre défini par Plotkin et qui construit un atome qui est leur plus petite généralisation. Cette notion est duale à celle de l'unification qui représente la spécialisation la plus générale entre deux atomes. Le **lgg** de deux graphes est le graphe couvrant de leur saturation. Nous avons prouvé que le temps du calcul d'un **lgg** ainsi que sa taille sont au pire quadratique par rapport à la taille des graphes RDF (saturées) en entrée.

Trouver les points communs entre les requêtes SPARQL

Pour définir le **lgg** entre des requêtes, nous utilisons les requêtes Basic Graph Pattern Queries (BGPQ), le fragment conjonctive de SPARQL, comme descriptions et l'implication standard entre BGPQs comme relation de spécialisation/généralisation. Dans ce cas, la définition du **lgg** a un intérêt pratique limité comme le montre le scénario qui suit. Soit deux requêtes : $q_1 \leftarrow (x, \tau, ConfPaper), (x, hasContactAuthor, y_1)$ qui cherche les articles de conférence qui ont un auteur de contact, et $q_2 \leftarrow (x, \tau, JourPaper), (x, hasAuthor, y)$ qui cherche les articles de journaux qui ont un auteur. Leur **lgg** en considérant l'implication standard est la requête qui cherche les ressources qui ont un auteur : $q_{\mathbf{lgg}}(x) \leftarrow (x, \tau, y)$. Ce **lgg** est très éloigné sémantiquement des requêtes initiales q_1 et q_2 . Néanmoins, en considérant un ensemble de contraintes ontologiques externes, comme (i) avoir un auteur de contact est plus spécifique qu'avoir un auteur, (ii) les articles de conférence (resp. articles de journal) sont des articles, nous pouvons définir un **lgg** plus spécifique qui cherche les articles qui ont un auteur : $q_{\mathbf{lgg}}(x) \leftarrow (x, \tau, Publication), (x, hasAuthor, y)$. Afin de trouver cette requête, nous avons défini une nouvelle relation d'implication entre requête BGPQs qui tient compte d'un ensemble de règles \mathcal{R} et d'un ensemble de contraintes ontologiques \mathcal{O} , notée $\models_{\mathcal{R}, \mathcal{O}}$. Cette relation repose sur une relation nouvellement introduite de saturation de requête selon \mathcal{R} et \mathcal{O} . Celle-ci permet de compléter le corps d'une requête avec tous les triplets qui peuvent être dérivés depuis le corps de la

requête uniquement et à partir de \mathcal{O} .

Nous avons montré que la requête et sa saturation sont équivalentes d'un point de vue des deux notions; l'implication de requête et de réponses à une requête, et que l'opération d'implication selon \mathcal{R} et \mathcal{O} entre requêtes est bien fondée par rapport à ces deux notions.

Nous avons proposé une nouvelle définition du **lgg** fondée sur l'implication de requête selon un ensemble de règles \mathcal{R} et un ensemble de contraintes ontologique \mathcal{O} . Nous avons prouvé que le **lgg** d'un ensemble de requêtes BGPQs peut ne pas exister, et que s'il existe, il est unique par rapport à la relation d'implication entre BGPQs.

Algorithmes et expérimentations

Pour calculer le **lgg** entre des graphes RDF, nous avons proposé trois algorithmes. Le premier permet de construire des **lggs** de graphes, resp. de requêtes qui tiennent en mémoire. Pour les graphes qui ne tiennent pas en mémoire, nous proposons un deuxième algorithme qui s'appuie sur les systèmes de gestion de bases de données. Enfin pour les graphes de très grande taille, nous avons défini un troisième algorithme qui fait appel au modèle de programmation MapReduce pour calculer le **lgg**. Nous avons également proposé un algorithme qui permet de construire des **lggs** de requêtes BGPQs. Ces algorithmes exploitent un algorithme qui calcule la plus petite anti-unification entre deux triplets de graphes ou de requêtes BGPQs.

Les expérimentations que nous avons menées avaient pour objectif de valider nos approches et de montrer la valeur ajoutée qu'apportent les contraintes ontologies dans le calcul du **lgg** de graphes RDF et de requêtes BGPQs. Pour cela, nous avons extrait un ensemble de graphes issues des données réelles de DBpedia, et défini des requêtes sur DBpedia et sur des données synthétiques LUBM représentant des universités.

Nous avons défini deux métriques pour mesurer le gain en précision apporté par l'utilisation de l'implication standard entre les graphes ($\models_{\mathcal{R}}$) et respectivement, par l'utilisation de l'implication entre requêtes BGPQs en présence de contraintes ontologiques ($\models_{\mathcal{R},\mathcal{O}}$) plutôt que l'implication simple (\models) – utilisée dans l'état de l'art – dans le calcul des **lggs** de graphes et de requêtes respectivement.

Nous avons calculé ces différents **lggs** et les gains de précision respectifs. Nos résultats confirment nos hypothèses : l'utilisation de l'implication standard entre les graphes RDF ($\models_{\mathcal{R}}$) et de l'implication entre BGPQs en considérant un ensemble de contraintes ontologiques ($\models_{\mathcal{R},\mathcal{O}}$) produit des **lggs** plus précis. En effet, l'exploitation des contraintes ontologiques permet de trouver les *généralisants communs* entre les classes et les propriétés qui sont utilisés à la place des classes et des propriétés des graphes et des requêtes initiaux. En utilisant l'implication simple, ces classes et ces propriétés ne peuvent être remplacées par des noeuds blancs dans les graphes et par des variables dans le cas des requêtes, ce qui engendre des très (trop) grandes généralisations, comme nos expérimentations le montrent.

État de l'art et conclusion

L'objectif de cette thèse est de revisiter le problème de la recherche des points communs entre des graphes RDF et des requêtes SPARQL, les deux standards populaires du web sémantique, et cela en calculant leur plus petit généralisant communs. Ce problème est très connu dans le domaine du Machine Learning [Plotkin, 1970, Plotkin, 1971, Buntine, 1988, Nienhuys-Cheng and de Wolf, 1996, Nienhuys-Cheng and de Wolf, 1997] et a été étudié dans d'autres domaines, comme les logiques des descriptions [Küstners, 2001, Baader et al., 2007, Zarriß and Turhan, 2013], les graphes conceptuels [Chein and Mugnier, 2009] et récemment dans RDF [Colucci et al., 2013, Colucci et al., 2016] et SPARQL [Lehmann and Böhmann, 2011, Böhmann et al., 2016]).

Les contributions de cette thèse reposent sur des idées développées en ILP pour calculer le **l_{gg}** des clauses du premier ordre [Plotkin, 1970, Plotkin, 1971, Buntine, 1988, Nienhuys-Cheng and de Wolf, 1996, Nienhuys-Cheng and de Wolf, 1997], dans le cadre de RDF et son langage de requête SPARQL. Les contributions de cette thèse améliorent l'état de l'art dans le domaine du web sémantique car nous considérons les graphes RDF et les BGPQs dans leur généralité, i.e., nous n'imposons aucune restriction ni structurelle ni sémantique sur les graphes et les requêtes en entrée.

Dans cette thèse nous donnons la définition, les caractéristiques et le calcul d'un **l_{gg}** de graphes RDF et de requêtes BGPQs (fragment conjonctif de SPARQL). Nous utilisons l'implication standard entre les graphes RDF ($\models_{\mathcal{R}}$) pour calculer un **l_{gg}** des graphes RDF et une extension de l'implication standard entre les BGPQs dotée d'un ensemble des contraintes ontologiques ($\models_{\mathcal{R}, \mathcal{O}}$) pour calculer un **l_{gg}** des BGPQs. Pour calculer le **l_{gg}** de graphes, nous proposons trois algorithmes pour gérer des graphes de petite taille, de grande taille, ou de très grande taille avec une approche dans laquelle les graphes tiennent (i) en mémoire, (ii) dans des systèmes de gestion de bases de données ou (iii) dans uncluster MapReduce. Les requêtes et leurs **l_{ggs}** tenant en mémoire, l'approche considérée pour le calcul des **l_{ggs}** entre des requêtes étend le premier algorithme (i).

Pour valider nos approches, nous avons effectué des expérimentations pour montrer le gain de précision qu'apporte une ontologie dans le calcul des **l_{ggs}**.

En termes de perspectives, nous envisageons d'étudier des techniques pour éliminer la redondance des **l_{ggs}** que nous calculons, il s'agirait de définir des heuristiques afin de calculer des **l_{ggs}** plus compactes. Par ailleurs, nous souhaitons étudier le problème de la recherche des points communs dans OWL2 QL, le deuxième standard du web sémantique recommandé par W3C qui correspond à la logique de description *DL-Lite_R*.

Introduction

Finding the commonalities between descriptions of data or knowledge is a foundational reasoning problem of Machine Learning, which was formalized by G. Plotkin in the early 70's as computing a *least general generalization* (**lgg**) of such descriptions [Plotkin, 1971, Plotkin, 1970]. This seminal work had been the basis of many research efforts in Machine Learning, in particular in its Inductive Logic Programming (ILP) subfield [Buntine, 1988, Nienhuys-Cheng and de Wolf, 1996, Nienhuys-Cheng and de Wolf, 1997].

Since the early 90's, this problem has also attracted the attention of the Knowledge Representation field, where the notion of least general generalization was rebaptized *least common subsumers* [Cohen et al., 1992]. In particular, it has been studied in Description Logics, e.g., [Cohen et al., 1992, Baader et al., 1999, Baader et al., 2007, Küsters, 2001, Zarri  and Turhan, 2013], and in Conceptual Graphs [Chein and Mugnier, 2009].

Also, more recently, this problem has started being investigated in the Semantic Web field, for the Resource Description Framework (RDF) [Colucci et al., 2016, El Hassad et al., 2017a] and its associated SPARQL query language [Lehmann and B hmann, 2011, B hmann et al., 2016, El Hassad et al., 2017e, El Hassad et al., 2017b], the two prominent and now well-established Semantic Web standards by W3C.

Motivations. In this thesis, we revisit the problem of computing an **lgg** of input descriptions in the Semantic Web setting (contributions to be outlined shortly), when these descriptions are either *RDF graphs* (i.e., datasets) or *SPARQL* queries, for which the literature provides limited solutions (limitations to be pointed out shortly).

Solutions to this problem can be applied to a variety of useful important applications, ranging from *query optimization* to *exploration* and *recommendation* in RDF data management systems or in SPARQL endpoints, as exemplified next.

In *Query optimization*, an **lgg** of incoming queries characterizes the largest set of their commonalities whose processing may be shared in *multi-query optimization* [Le et al., 2012]. Similarly, **lggs** of subsets of a query workload correspond to candidate views that may be recommended for materialization in *view selection* [Goasdou  et al., 2011], a typical optimization for data warehouses [Colazzo et al., 2014], and among which can be selected those that allow rewrit-

ing (partially or totally) the workload while minimizing a combination of rewriting processing, view storage and view maintenance costs.

In *Recommendation*, in particular in a social context, **lggs** of sets of user descriptions (i.e., profiles) may help recommending users other users, or to form a community, when they share enough interests [Cheng et al., 2014, Lohmann et al., 2010]. Also, **lggs** of sets of queries issued by distinct users may help recommending users to each other, if what they ask for is enough related [Huang et al., 2016].

In *Exploration*, **lggs** of datasets may be used to classify/categorize them w.r.t. their common information [Grimnes et al., 2008], to identify common social graph patterns between organizations [Jin et al., 2005, Perner, 2017] (e.g., criminal ones), or to help identifying new potential links between datasets in the Linked Data Cloud [Sherif et al., 2017]. *Clustering user queries* found in system logs, based on their **lggs**, may help classifying the queries and identifying the kind of data each category accesses [Chuang and Chien, 2002]. Further, finding the relevant user query cluster for an incoming query may help recommending *similar and complementary searches* [Huang et al., 2016, Heckel et al., 2017].

Contributions. The contribution of this thesis to the problem of computing a least general generalizations in the RDF and in SPARQL queries are as follows:

1. We define and study the problem of computing an **lgg** of RDF graphs in the *entire RDF standard*: we do not restrict RDF graphs in any way, i.e., neither their structure nor their semantics defined upon RDF entailment (inference). The recent work [Colucci et al., 2013, Colucci et al., 2016] brings a limited solution to the problem. It allows finding the commonalities between *single entities* extracted from RDF graphs (e.g., users in a social network), *ignoring* RDF entailment. In contrast, we further aim at considering the problem in all its generality, i.e., finding the commonalities between *general* RDF graphs, hence modeling *multiple interrelated entities* (e.g., social networks of users), faithfully w.r.t. their standard semantics. This work has been published in [El Hassad et al., 2017a] and presented to the ILP community [El Hassad et al., 2017b].
2. We define and study the problem of computing an **lgg** of *general* SPARQL conjunctive queries, a.k.a. Basic Graph Pattern Queries (BGPQs), while the literature only considers unary tree-shaped conjunctive queries (UTCQ) [Lehmann and Böhmann, 2011]. Further, when available, we devise how to use background knowledge, formalized as ontological constraints modeling the application domain, in order to compute much more precise **lggs**. This work has been published in [El Hassad et al., 2017e, El Hassad et al., 2017c] and presented to the french database community [El Hassad et al., 2017d].
3. We provide algorithms for our solutions to these two problems. In particular, since RDF graphs may be large, we provide algorithms that allow computing **lggs** of *small-to-huge general RDF graphs* (i.e., that fit either in memory, in data management systems or in MapReduce clusters) w.r.t. *any set of entailment rules* from the RDF standard. Our algorithms for RDF graphs have been

published in [El Hassad et al., 2017a].

4. We report on experiments using DBpedia, a *real* dataset, which demonstrate the added-value of considering standard entailment when learning **lggs** between RDF graphs. We also report on experiments using *synthetic* data (LUBM) and *real* data (DBpedia), which show to which extent **lggs** of BGPQs are much more precise when standard entailment endowed with background knowledge is used. To this aim, we define metrics to compute the gain in precision using RDF entailment yields.

Organization This thesis is organized as follows. In Chapter 1, we introduce the RDF data model and its SPARQL query language. Then, we study the problem of computing an **lgg** of RDF graphs in Chapter 2, and of BGPQs in Chapter 3. In Chapter 4, we present algorithms for the two aforementioned problems and we report on the experiments we conducted to evaluate our technical contributions in Chapter 5. We present related works in Chapter 6. Finally, we conclude and draw some perspectives in Chapter 7.

Chapter 1

Preliminaries

Contents

1.1	Introduction	17
1.2	Resource Description Framework (RDF)	17
1.2.1	RDF graphs	17
1.2.2	Adding ontological knowledge to RDF graphs	18
1.2.3	Deriving the implicit triples of an RDF graph	19
1.2.4	Comparing RDF graphs	21
1.3	SPARQL conjunctive queries	21
1.3.1	Basic graph pattern queries	21
1.3.2	Entailing and answering queries	22
1.3.3	Comparing queries	23
1.4	Conclusion	24

1.1 Introduction

In this chapter we recall the basics of the prominent W3C’s Semantic Web standards: the RDF data model in Section 1.2, and its associated SPARQL query language in Section 1.3.

1.2 Resource Description Framework (RDF)

1.2.1 RDF graphs

The RDF data model allows specifying *RDF graphs*. An RDF graph is a set of *triples* of the form (s, p, o) . A triple states that its *subject* s 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

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

Rule	Entailment rule
rdfs2	$(p, \leftrightarrow_d, o), (s_1, p, o_1) \rightarrow (s_1, \tau, o)$
rdfs3	$(p, \leftrightarrow_r, o), (s_1, p, o_1) \rightarrow (o_1, \tau, o)$
rdfs5	$(p_1, \preceq_{sp}, p_2), (p_2, \preceq_{sp}, p_3) \rightarrow (p_1, \preceq_{sp}, p_3)$
rdfs7	$(p_1, \preceq_{sp}, p_2), (s, p_1, o) \rightarrow (s, p_2, o)$
rdfs9	$(s, \preceq_{sc}, o), (s_1, \tau, s) \rightarrow (s_1, \tau, o)$
rdfs11	$(s, \preceq_{sc}, o), (o, \preceq_{sc}, o_1) \rightarrow (s, \preceq_{sc}, o_1)$
ext1	$(p, \leftrightarrow_d, o), (o, \preceq_{sc}, o_1) \rightarrow (p, \leftrightarrow_d, o_1)$
ext2	$(p, \leftrightarrow_r, o), (o, \preceq_{sc}, o_1) \rightarrow (p, \leftrightarrow_r, o_1)$
ext3	$(p, \preceq_{sp}, p_1), (p_1, \leftrightarrow_d, o) \rightarrow (p, \leftrightarrow_d, o)$
ext4	$(p, \preceq_{sp}, p_1), (p_1, \leftrightarrow_r, o) \rightarrow (p, \leftrightarrow_r, o)$

Table 1.1 – RDF & RDFS statements. Table 1.2 – Sample RDF entailment rules [W3C-RDFS, 2014].

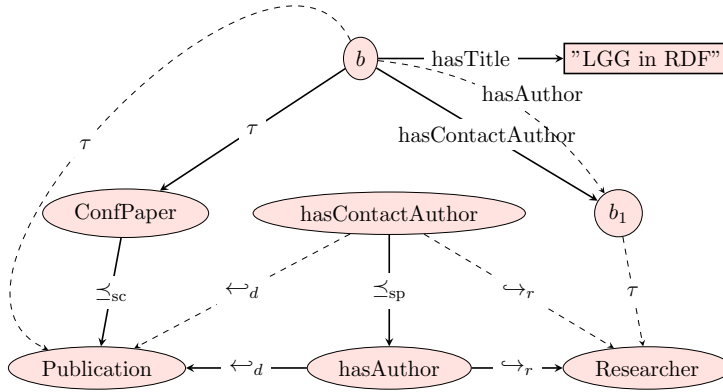
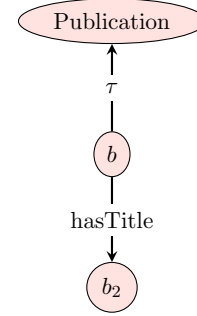
specification [W3C-RDF, 2014], belong to $(\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{L} \cup \mathcal{B})$; we only consider such triples hereafter.

Notations. We use s, p, o in triples as placeholders. We note $\text{Val}(\mathcal{G})$ the set of *values* occurring in an RDF graph \mathcal{G} , i.e., the URIs, literals and blank nodes; we note $\text{Bl}(\mathcal{G})$ 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 1.1 (top) shows the use of triples to state such assertions. The RDF standard [W3C-RDF, 2014] 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 $(b, \text{rdf:type}, \text{ConfPaper})$.

1.2.2 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 built-in properties. Table 1.1 (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 $(\text{ConfPaper}, \text{rdfs:subClassOf}, \text{Publication})$ 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 may have authors*, and the triple $(\text{hasAuthor}, \text{rdfs:range}, \text{Researcher})$ states that *only researchers may be authors of something*.

Figure 1.1 – Sample RDF graph \mathcal{G} .Figure 1.2 – Sample RDF graph \mathcal{G}' .

Notations. For conciseness, we use the following shorthands for built-in properties: τ for `rdf:type`, \preceq_{sc} for `rdfs:subClassOf`, \preceq_{sp} for `rdfs:subPropertyOf`, \leftrightarrow_d for `rdfs:domain`, and \leftrightarrow_r for `rdfs:range`.

Figure 1.1 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 (s, p, o) corresponds to a p -labeled directed edge from the s node to the o node: $s \xrightarrow{p} o$. 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 Figure 1.1, the constraint $(\text{hasContactAuthor}, \preceq_{sp}, \text{hasAuthor})$ together with the triple $(b, \text{hasContactAuthor}, b_1)$ imply the implicit triple $(b, \text{hasAuthor}, b_1)$, which, further, with the constraint $(\text{hasAuthor}, \leftrightarrow_r, \text{Researcher})$ yields another implicit triple $(b_1, \tau, \text{Researcher})$.

1.2.3 Deriving the implicit triples of an RDF graph

The implicit triples of an RDF graph are derived with the help of *entailment rules* from the RDF standard [W3C-RDFS, 2014].

An *entailment rule* r is of the form $body(r) \rightarrow head(r)$, where $body(r)$ is an RDF graph called the *body* of r and $head(r)$ is also an RDF graph, called the *head* of r ; it states that the RDF graph $body(r)$ entails the RDF graph $head(r)$. Table 1.2 shows the strict subset of these rules that we will use to illustrate important notions as well as our contributions in the next chapters. Crucially, our contributions hold for the entire set of entailment rules of the RDF standard, and any subset of thereof. The rules in Table 1.2 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 \preceq_{sp} and \preceq_{sc} constraints (`rdfs5`,

`rdfs11`), the *complementation* of domains or ranges through \preceq_{sc} (`ext1`, `ext2`), and the *inheritance* of domains and of ranges through \preceq_{sp} (`ext3`, `ext4`).

An entailment rule r can be applied to an RDF graph \mathcal{G} to identify some \mathcal{G} implicit triple(s) if \mathcal{G} *simply entails* $body(r)$, i.e., \mathcal{G} contains some particular case(s) of $body(r)$. *Simple entailment between RDF graphs* is the generalization/specialization relation from the RDF standard [W3C-RDFS, 2014] that allows comparing RDF graphs based on their *explicit* triples *only*. More formally, given two RDF graphs \mathcal{G} and \mathcal{G}' , \mathcal{G} *simply entails* \mathcal{G}' , denoted $\mathcal{G} \models \mathcal{G}'$, if there exists a homomorphism ϕ from the blank nodes of \mathcal{G}' to \mathcal{G} values (URIs, blank nodes and literals), i.e., from $\text{Bl}(\mathcal{G}')$ to $\text{Val}(\mathcal{G})$, such that $[\mathcal{G}']_{\phi} \subseteq \mathcal{G}$, where $[\mathcal{G}']_{\phi}$ is the RDF graph obtained from \mathcal{G}' by replacing every blank node b with its image $\phi(b)$.

When an RDF graph \mathcal{G} simply entails the body of an entailment rule r due to some homomorphism ϕ , in which case we say that \mathcal{G} *triggers* (a.k.a. *fires*) r due to ϕ , this rule allows deriving that \mathcal{G} entails $[head(r)]_{\phi}$, because $[body(r)]_{\phi} \subseteq \mathcal{G}$ holds and clearly $[body(r)]_{\phi} \rightarrow [head(r)]_{\phi}$ is a particular case of the rule r .

Notation. From now, we denote by $\mathcal{G} \models^{\phi} \mathcal{G}'$ the fact that the RDF graph \mathcal{G}' is simply entailed by the RDF graph \mathcal{G} due to the homomorphism ϕ .

Given a set \mathcal{R} of entailment rules from the RDF standard, a.k.a. entailment regime, the *saturation* (or *closure*) of an RDF graph \mathcal{G} w.r.t. \mathcal{R} is the RDF graph \mathcal{G}^{∞} obtained by adding to \mathcal{G} *all* the implicit triples that follow from \mathcal{G} and \mathcal{R} . The saturation \mathcal{G}^{∞} *materializes* the semantics of \mathcal{G} w.r.t. \mathcal{R} . 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 new triples they derive. More formally, given an RDF graph \mathcal{G} and a set \mathcal{R} of entailment rules, the saturation of \mathcal{G} with \mathcal{R} is the fixpoint \mathcal{G}^{∞} recursively defined as:

- $\mathcal{G}_0 = \mathcal{G}$
- $\mathcal{G}_n = \mathcal{G}_{n-1} \cup \{[head(r)]_{\phi} \mid r \in \mathcal{R} \wedge \mathcal{G}_{n-1} \models^{\phi} body(r) \wedge \mathcal{G}_{0 \leq i < n-1} \not\models^{\phi} body(r)\}$, such that if a blank node b in $head(r)$ has no image through ϕ , then $\phi(b)$ is a *fresh* blank node.

Further, for any RDF graph and set of entailment rules from the RDF standard, the following holds [W3C-RDFS, 2014]:

Property 1. *The saturation is finite, unique (up to blank node renaming), and can be computed in polynomial time*¹.

The interested reader may find, for particular sets of entailment rules from the RDF standard, the upper bounds of worst-case saturation time (polynomials of low degrees) and of saturation size in e.g., [ter Horst, 2005, Goasdoué et al., 2013].

1. This may seem counter-intuitive at first glance, because RDF graph saturation builds on simple entailment between RDF graph, which is NP-complete. The polynomial complexity follows from the fact that the set of entailment rules from the RDF standard is *fixed*, hence the size (number of triples) in the bodies of these rules is bounded by some constant.

The saturation of the RDF graph \mathcal{G} shown in Figure 1.1 corresponds to the RDF graph \mathcal{G}^∞ 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}^∞ . 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: $(\text{hasContactAuthor}, \preceq_{\text{sp}}, \text{hasAuthor})$ 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 $(\text{hasAuthor}, \hookrightarrow_\tau, \text{Researcher})$ triggers **rdfs3** that adds $(b_1, \tau, \text{Researcher})$.

1.2.4 Comparing RDF graphs.

The RDF standard defines a generalization/specialization relationship between two RDF graphs, called *entailment between graphs*, that goes beyond simple entailment by taking into account the explicit *and* the implicit triples of the compared RDF 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 a graph homomorphism from \mathcal{G}' to the *saturation* of \mathcal{G} , i.e., the complete set of triples that \mathcal{G} models. 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 $\text{Bl}(\mathcal{G}')$ to $\text{Val}(\mathcal{G}^\infty)$ 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 $\phi(b)$. Further, deciding entailment between two RDF graphs is NP-complete [W3C-RDFS, 2014]; this follows from (i) the fact that the saturation of \mathcal{G} can be computed in polynomial time and (ii) from the NP-completeness of deciding whether there exists a homomorphism between two graphs.

Figure 1.2 shows an RDF graph \mathcal{G}' entailed by the RDF graph \mathcal{G} in Figure 1.1 w.r.t. the entailment rules displayed in Table 1.2. In particular, $\mathcal{G} \models_{\mathcal{R}} \mathcal{G}'$ holds for the homomorphism ϕ such that: $\phi(b) = b$ and $\phi(b_2) = \text{"LGG in SPARQL"}$. 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 $\mathcal{G} \models_{\mathcal{R}}^\phi \mathcal{G}'$ the fact that the entailment $\mathcal{G} \models_{\mathcal{R}} \mathcal{G}'$ holds due to the graph homomorphism ϕ .

Importantly, from the definition of entailment between two RDF graphs [W3C-RDF, 2014], the following holds:

- Property 2.** *Given two RDF graphs $\mathcal{G}, \mathcal{G}'$ and a set \mathcal{R} of RDF entailment rules,*
1. \mathcal{G} and \mathcal{G}^∞ are equivalent ($\mathcal{G} \models_{\mathcal{R}} \mathcal{G}^\infty$ and $\mathcal{G}^\infty \models_{\mathcal{R}} \mathcal{G}$ hold), noted $\mathcal{G} \equiv_{\mathcal{R}} \mathcal{G}^\infty$,
 2. $\mathcal{G} \models_{\mathcal{R}} \mathcal{G}'$ holds iff $\mathcal{G}^\infty \models \mathcal{G}'$ holds.

From a practical viewpoint, Property 2 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}^∞ of \mathcal{G} , followed by a standard graph homomorphism step that checks if $\mathcal{G}^\infty \models \mathcal{G}'$ holds.

Finally, we remark that when entailment rules are not considered, i.e., $\mathcal{R} = \emptyset$, the relations of simple entailment and of (general) entailment coincide.

1.3 SPARQL conjunctive queries

1.3.1 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 select-project-join queries for databases; it is the most widely used subset of SPARQL queries in real-world applications [Picalausa et al., 2012].

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 $q(\bar{x}) \leftarrow t_1, \dots, t_\alpha$, where $\{t_1, \dots, t_\alpha\}$ 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 $\{t_1, \dots, t_\alpha\}$, 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, \dots, t_α ; 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 $\text{VarBl}(q)$ the set of variables *and* blank nodes occurring in the query q , and $\text{Val}(q)$ the set of all its values, i.e., URIs, blank nodes, literals and variables.

1.3.2 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 $\text{VarBl}(q)$ to $\text{Val}(\mathcal{G}^\infty)$ such that $[body(q)]_\phi \subseteq \mathcal{G}^\infty$. Further, similarly to entailment between RDF graphs (and for the same reasons), deciding query entailment is NP-complete [W3C-SPARQL, 2008].

The RDF graph \mathcal{G} in Figure 1.1 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 Figure 1.1.

Notations. Similarly to entailment between RDF graphs, we denote by $\mathcal{G} \models_{\mathcal{R}}^{\phi} q$ 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

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

where $(\bar{x})_{\phi}$ is the tuple of \mathcal{G}^{∞} 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. Further, because query entailment is NP-complete, clearly, identifying an answer of a BGPQ against an RDF graph is NP-complete [W3C-SPARQL, 2008].

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 1.1 is:

- $\{\langle b, \text{ConfPaper} \rangle\}$ for $\mathcal{R} = \emptyset$, i.e., considering only the explicit triples in Figure 1.1;
- $\{\langle b, \text{ConfPaper} \rangle, \langle b, \text{Publication} \rangle, \langle b_1, \text{Researcher} \rangle\}$ for \mathcal{R} the set of entailment rules in Table 1.2, i.e., considering the explicit *and* implicit triples in Figure 1.1.

Importantly, from the definition of answer set of a SPARQL query against an RDF graph [W3C-SPARQL, 2008], the following holds:

Property 3. *Given an RDF graph \mathcal{G} , a set \mathcal{R} of entailment rules and a BGPQ q ,*

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

From a practical viewpoint, Property 3 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}^{∞} 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.

1.3.3 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 $body(q) \models_{\mathcal{R}}^{\phi} body(q')$ with $(\bar{x}')_{\phi} = \bar{x}$ holds. Here, $body(q) \models_{\mathcal{R}}^{\phi} body(q')$ 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 $\text{VarBl}(body(q'))$ to $\text{Val}(body(q)^{\infty})$ 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 [W3C-SPARQL, 2008]. Similarly to entailment between RDF graphs (and for the same reasons), deciding entailment between two BGPQs is NP-complete [W3C-RDFS, 2014, W3C-SPARQL, 2008].

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 4. *Given an RDF graph \mathcal{G} , a set \mathcal{R} of entailment rules and two BGPQs q, q' such that $q \models_{\mathcal{R}} q'$,*

1. *if $\mathcal{G} \models_{\mathcal{R}} q$ holds then $\mathcal{G} \models_{\mathcal{R}} q'$ holds,*
2. *$q(\mathcal{G}) \subseteq q'(\mathcal{G})$ holds.*

Finally, query entailment, query answering and entailment between queries *treat blank nodes in queries exactly as non-answer variables* [W3C-SPARQL, 2008]. Hence, hereafter, we assume *without loss of generality* that queries do not use blank nodes.

1.4 Conclusion

In this chapter, we recalled the RDF and SPARQL notions on which the main contributions of this PhD thesis build: the definition and computation of the largest set of commonalities between RDF graphs or between BGPQs, which we present in the next two chapters.

Chapter 2

Finding Commonalities between RDF graphs

Contents

2.1	Introduction	27
2.2	Defining the lgg of RDF graphs	28
2.3	Computing an lgg of RDF graphs	31
2.4	Conclusion	35

2.1 Introduction

In this chapter, we present the first main contribution of this thesis: we define and solve the problem of learning the commonalities between RDF graphs, formalized as a least general generalization (l g g for short) of these RDF graphs w.r.t. the standard relation of entailment between RDF graphs.

In Section 2.2, we define and study the l g g of input RDF graphs, as well as how to compute it and the resources needed for that in Section 2.3. Such an l g g is an RDF graph that represents the largest set of commonalities of the input RDF graphs. Crucially, and in contrast with the state of the art (see Chapter 6), our approach is faithful to the entire W3C standard: we do not restrict the structure of the input/output RDF graphs in any way, nor their semantics, which is based on RDF entailment.

2.2 Defining the lgg of RDF graphs

A *least general generalization* of n descriptions d_1, \dots, d_n is a most specific description d generalizing every $d_{1 \leq i \leq n}$ for some generalization/specialization relation between

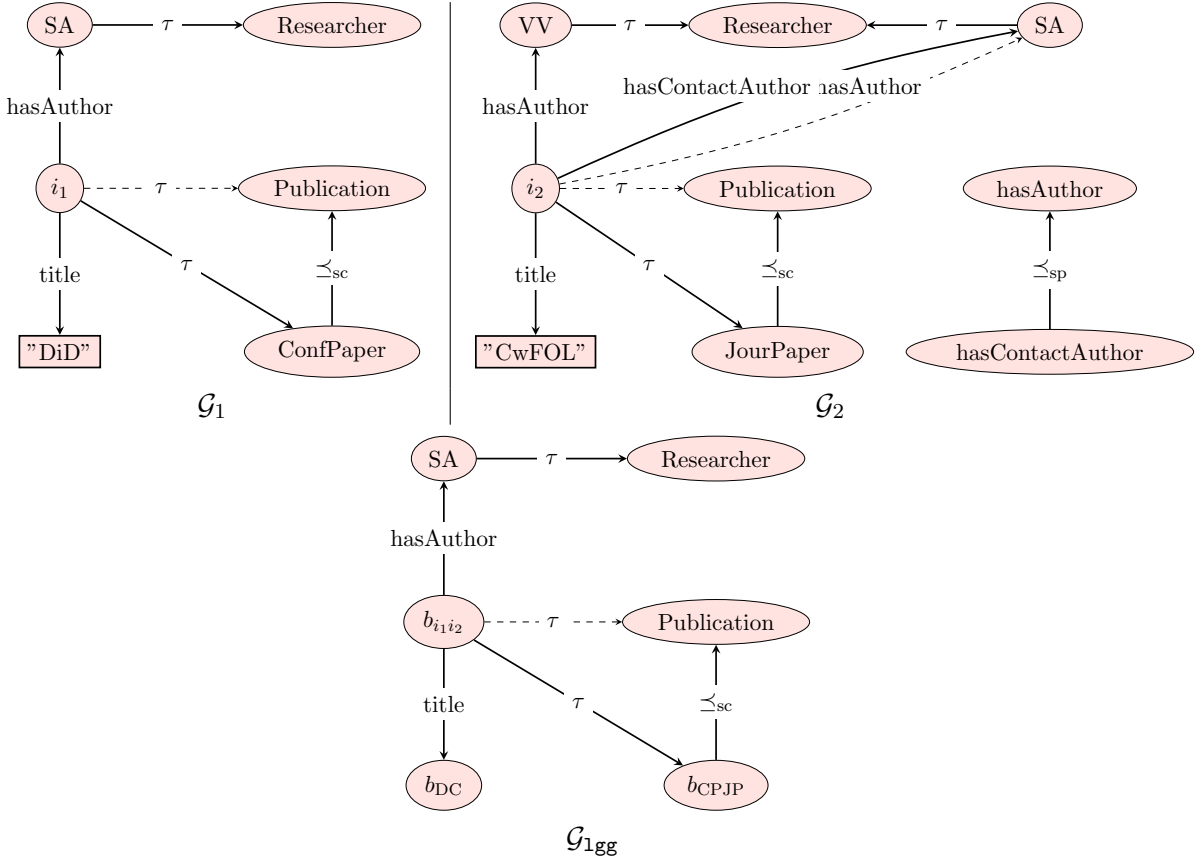


Figure 2.1 – Sample RDF graphs \mathcal{G}_1 , \mathcal{G}_2 and \mathcal{G}_{1gg} , with \mathcal{G}_{1gg} the minimal 1gg of \mathcal{G}_1 and \mathcal{G}_2 ; their implicit triples (i.e., derived by the rules in Table 1.2) are shown as dashed edges.

descriptions [Plotkin, 1970, Plotkin, 1971]. In RDF, we use *RDF graphs* as descriptions and *entailment between RDF graphs* as relation for generalization/specialization:

Definition 1 (1gg of RDF graphs). *Let $\mathcal{G}_1, \dots, \mathcal{G}_n$ be RDF graphs and \mathcal{R} a set of RDF entailment rules.*

- A generalization of $\mathcal{G}_1, \dots, \mathcal{G}_n$ 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 (1gg) of $\mathcal{G}_1, \dots, \mathcal{G}_n$ is a generalization \mathcal{G}_{1gg} of $\mathcal{G}_1, \dots, \mathcal{G}_n$ such that for any other generalization \mathcal{G}_g of $\mathcal{G}_1, \dots, \mathcal{G}_n$, $\mathcal{G}_{1gg} \models_{\mathcal{R}} \mathcal{G}_g$ holds.

Importantly, in the RDF setting, the following holds:

Theorem 1. *An 1gg of RDF graphs always exists; it is unique up to entailment.*

Proof. An 1gg of RDF graphs always exists, since we can always construct a (possibly empty) RDF graph that is the 1gg of RDF graphs, in particular the *cover graph of RDF graphs* devised in Section 2.3.

Also, an lgg of RDF graphs is *unique* up to entailment (since $\mathcal{G}_{\text{lgg}} \models_{\mathcal{R}} \mathcal{G}_g$ holds for any \mathcal{G}_g in Definition 1). Indeed, if it were that RDF graphs have *multiple* lgg's *incomparable* w.r.t. entailment, say $\text{lgg}_1, \dots, \text{lgg}_m$, their merge¹ $\text{lgg}_1 \uplus \dots \uplus \text{lgg}_m$ would be a *single strictly more specific* lgg, a contradiction. \square

Figure 2.1 displays two RDF graphs \mathcal{G}_1 and \mathcal{G}_2 , as well as their minimal lgg \mathcal{G}_{lgg} (with lowest number of triples), when we consider the RDF entailment rules shown in Table 1.2. \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}_{lgg} states that their common information comprises the existence of a resource ($b_{i_1 i_2}$) having some type ($b_{C(\text{onf})P(\text{aper})J(\text{our})P(\text{aper})}$), which is a particular case of publication, with some title ($b_{D(\text{id})C(\text{wFOL})}$) 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 at hand allows deriving them from the remaining triples (e.g., materializing the only \mathcal{G}_{lgg} implicit triple in Figure 2.1 would make it redundant if we consider the entailment rules in Table 1.2) or triples generalizing others without needing RDF entailment rules, i.e., w.r.t. \models_{\emptyset} (e.g., adding the triple $(b, \text{hasAuthor}, b')$ to \mathcal{G}_{lgg} in Figure 2.1 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 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., [Meier, 2008, Pichler et al., 2010, Pichler et al., 2013]. Hence we focus in this work on the following learning problem:

Problem 1. *Given the RDF graphs $\mathcal{G}_1, \dots, \mathcal{G}_n$ and a set \mathcal{R} of RDF entailment rules, we want to compute some lgg of $\mathcal{G}_1, \dots, \mathcal{G}_n$.*

The proposition below states that an lgg of n RDF graphs, with $n \geq 3$, can be inductively defined (hence computed) as a sequence of $n - 1$ lgg's of *two* RDF graphs.

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 (Chapter 1).

Intuitively, assuming that $\ell_{k \geq 2}$ is an operator computing an **lgg** of k input RDF graphs, the next proposition establishes that:

$$\begin{aligned} [\text{basis}] \quad & \ell_3(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3) \equiv_{\mathcal{R}} \ell_2(\ell_2(\mathcal{G}_1, \mathcal{G}_2), \mathcal{G}_3) \\ [\text{induction}] \quad & \ell_n(\mathcal{G}_1, \dots, \mathcal{G}_n) \equiv_{\mathcal{R}} \ell_2(\ell_{n-1}(\mathcal{G}_1, \dots, \mathcal{G}_{n-1}), \mathcal{G}_n) \\ & \equiv_{\mathcal{R}} \ell_2(\ell_2(\dots \ell_2(\ell_2(\mathcal{G}_1, \mathcal{G}_2), \mathcal{G}_3) \dots, \mathcal{G}_{n-1}), \mathcal{G}_n) \end{aligned}$$

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}_{lgg} is an **lgg** of $\mathcal{G}_1, \dots, \mathcal{G}_n$ iff \mathcal{G}_{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 $\mathcal{G}_{\text{lgg}}^1$ is an **lgg** of $\mathcal{G}_1, \dots, \mathcal{G}_{k < n}$ and $\mathcal{G}_{\text{lgg}}^2$ is an **lgg** of $\mathcal{G}_1, \dots, \mathcal{G}_n$, then $\mathcal{G}_{\text{lgg}}^1 \models_{\mathcal{R}} \mathcal{G}_{\text{lgg}}^2$ holds.*

Let us show that the above lemma holds.

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

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

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

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

(\Rightarrow) Assume that \mathcal{G}_{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}_{lgg} is an **lgg** of $\mathcal{G}_1, \dots, \mathcal{G}_n$, we have: (i) $\mathcal{G}_{1 \leq i \leq n} \models_{\mathcal{R}} \mathcal{G}_{\text{lgg}}$ holds and (ii) for any RDF graph \mathcal{G} such that $\mathcal{G}_{1 \leq i \leq n} \models_{\mathcal{R}} \mathcal{G}$ holds, $\mathcal{G}_{\text{lgg}} \models_{\mathcal{R}} \mathcal{G}$ holds.

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

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

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

From Definition 1, (*) and (**) we get that \mathcal{G}_{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 particular instance of Problem 1 for $n = 2$.

2.3 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 3 below), which is central to our technique for computing an lgg of \mathcal{G}_1 and \mathcal{G}_2 . We indeed show (Theorem 2) 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 3): 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, as well as the worst-case time to compute them.

Our notion of cover graph builds on that of *least general anti-unification* of two first order atoms [Plotkin, 1970, Plotkin, 1971, Robinson and Voronkov, 2001], which defines an atom that is their least general generalization. This is the dual to the well-known notion of *most general unifier* of two first order atoms [Robinson, 1965, Robinson and Voronkov, 2001], which defines an atom that is their most general specialization. We transfer it to the RDF setting as follows:

Definition 2 (Anti-unification of triples). *Let ς be an injective generalization function from $(\mathcal{U} \cup \mathcal{L} \cup \mathcal{B}) \times (\mathcal{U} \cup \mathcal{L} \cup \mathcal{B})$ to $\mathcal{U} \cup \mathcal{L} \cup \mathcal{B}$ that maps (i) any pair of same input URI or literal value to itself, i.e., $\varsigma(v, v) = v$, and (ii) any other pair of values (URIs, blank nodes, literals and mix thereof) to a blank node.*

The anti-unification of the two triples (t_1, p, t_2) and (t_3, p, t_4) is the triple $(\varsigma(t_1, t_3), p, \varsigma(t_2, t_4))$.

Observe that anti-unification is only defined for triples with *same* property URI. Indeed, anti-unifying triples of the form (t_1, p, t_2) and (t_3, p', t_4) , with $p \neq p'$, would lead to a *non-well-formed* triples of the form (t_5, b, t_6) (recall that property values *must* be URIs in RDF graphs), where b is the blank node required to generalize the distinct values p and p' .

With the notion of anti-unification of triples in place, we are now able to define our central notion of cover graph:

Definition 3 (Cover graph). *The cover graph \mathcal{G} of two RDF graphs \mathcal{G}_1 and \mathcal{G}_2 is the RDF graph, which may be empty, 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 $b_{t_2 t_4}$.

It is worth noting that the definition of cover graph does not manipulate explicitly the injective generalization function ς . It is left implicitly in our way of defining the values of t_5 and t_6 : $t_5 = t_1$ if $t_1 = t_3$ and $t_1 \in \mathcal{U} \cup \mathcal{L}$ corresponds to the particular case $\varsigma(v, v) = v$ for v a URI or literal, while *otherwise* t_5 is the blank node $b_{t_1 t_3}$ corresponds to all the other cases, i.e., $t_5 = b_{t_1 t_3} = \varsigma(t_1, t_3)$; this is the same for t_6 .

Intuitively, the cover graph is a *generalization* of \mathcal{G}_1 and \mathcal{G}_2 (first item in Definition 1) as each of its triple (t_5, p, t_6) is a *least general anti-unification* of a triple (t_1, p, t_2) from \mathcal{G}_1 and a triple (t_3, p, t_4) from \mathcal{G}_2 . Further, the cover graph is an **lgg** for the explicit triples in \mathcal{G}_1 and those in \mathcal{G}_2 (second item in Definition 1) since 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 $b_{tt'}$ 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 example in Figure 2.1, the \mathcal{G}_1 *explicit* triples $(i_1, \tau, \text{ConfPaper})$, $(\text{ConfPaper}, \preceq_{\text{sc}}, \text{Publication})$, $(i_1, \text{title}, \text{"DiD"})$ and the \mathcal{G}_2 *explicit* triples $(i_2, \tau, \text{JourPaper})$, $(\text{JourPaper}, \preceq_{\text{sc}}, \text{Publication})$, $(i_2, \text{title}, \text{"CwFOL"})$, lead to the triples $(b_{i_1 i_2}, \tau, b_{\text{CPJP}})$, $(b_{\text{CPJP}}, \preceq_{\text{sc}}, \text{Publication})$, $(b_{i_1 i_2}, \text{title}, b_{\text{DC}})$ in the cover graph of \mathcal{G}_1 and \mathcal{G}_2 shown in Figure 2.2 (top). The first above-mentioned \mathcal{G} triple results from anti-unifying i_1 and i_2 into $b_{i_1 i_2}$, 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 $b_{i_1 i_2}$, and, "DiD" and "CwFOL" into b_{DC} . By reusing consistently the same blank node name $b_{i_1 i_2}$ for each anti-unification of the constants i_1 and i_2 (resp. b_{CPJP} for ConfPaper and JourPaper), the cover graph triples join on $b_{i_1 i_2}$ (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 2. *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 3 (see below), noting that in the particular case of Theorem 2, $\mathcal{R} = \emptyset$ holds, hence $\mathcal{G}_1^\infty = \mathcal{G}_1$ and $\mathcal{G}_2^\infty = \mathcal{G}_2$ holds. \square

We provide below worst-case bounds for the time to compute a cover graph and for its size; these bounds directly follows from the definition of a cover graph (Definition 3)

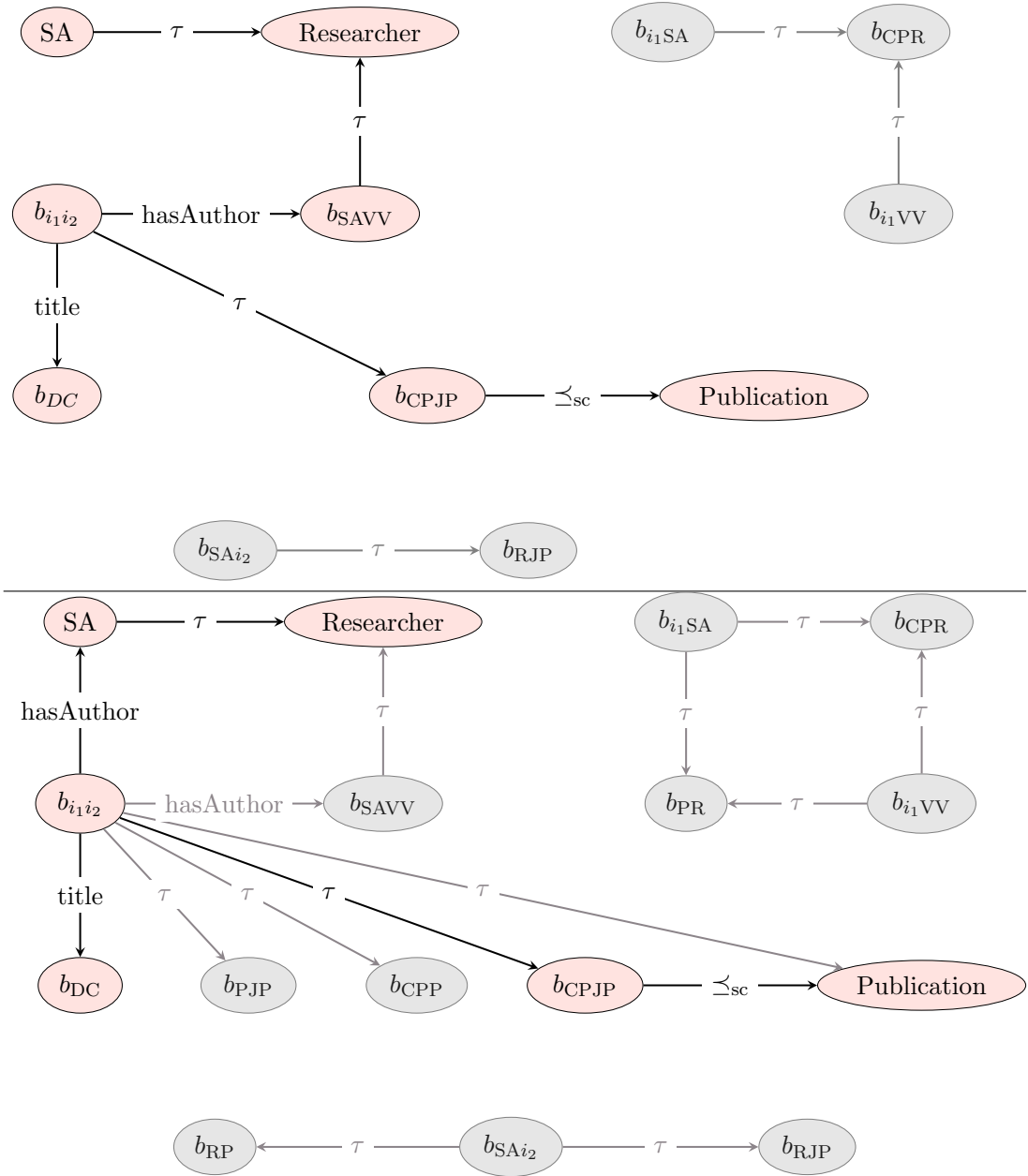


Figure 2.2 – Cover graphs of \mathcal{G}_1 and \mathcal{G}_2 in Figure 2.1 (top) and of their saturations w.r.t. the entailment rules in Table 1.2. Triples shown in gray are part of the graph but are redundant w.r.t. those shown in pink.

and are met when *all the triples of the two input graphs use the same property URI* (i.e., every pair of \mathcal{G}_1 and \mathcal{G}_2 triples begets a \mathcal{G} triple).

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 2 in order to take into account any set of entailment rules from the RDF standard. It states that it is sufficient to compute the cover graph of the saturations of the input RDF graphs, instead of the input RDF graphs themselves.

Theorem 3. *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^∞ and \mathcal{G}_2^∞ exists and is an **lgg** of \mathcal{G}_1 and \mathcal{G}_2 .*

Proof. Clearly, by definition, a cover graph \mathcal{G} of \mathcal{G}_1^∞ and \mathcal{G}_2^∞ always exists; it may be empty when the triples in \mathcal{G}_1^∞ and those in \mathcal{G}_2^∞ have no property URI in common.

Now, we show 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 $\mathcal{G}_2 \models_{\mathcal{R}} \mathcal{G}$. If \mathcal{G} is empty, it's trivial. Otherwise, consider the RDF graph \mathcal{G}' obtained from \mathcal{G} by replacing every blank node $b_{v_1 v_2}$ by the value v_1 . Clearly, $\mathcal{G}' \models_{\mathcal{R}} \mathcal{G}$ and, by construction of \mathcal{G} , $\mathcal{G}' \subseteq \mathcal{G}_1$, 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}_{lgg} be *any* **lgg** of \mathcal{G}_1 and \mathcal{G}_2 . To prove our claim, we need to show that $\mathcal{G} \models_{\mathcal{R}} \mathcal{G}_{\text{lgg}}$ holds. If \mathcal{G}_{lgg} is empty, then so is \mathcal{G} , since $\mathcal{G}_{\text{lgg}} \models_{\mathcal{R}} \mathcal{G}$ (\mathcal{G}_{lgg} is an **lgg** of \mathcal{G}_1 and \mathcal{G}_2 , and \mathcal{G} only a generalization of them). Otherwise, since \mathcal{G}_{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}_{lgg} to the values in \mathcal{G}_1^∞ and in \mathcal{G}_2^∞ respectively, such that $[\mathcal{G}_{\text{lgg}}]_{\phi_1} \subseteq \mathcal{G}_1^\infty$ and $[\mathcal{G}_{\text{lgg}}]_{\phi_2} \subseteq \mathcal{G}_2^\infty$. Consider the graph $\mathcal{G}'_{\text{lgg}}$ obtained from \mathcal{G}_{lgg} by replacing every blank node b by $b_{v_1 v_2}$ where $v_1 = \phi_1(b)$ and $v_2 = \phi_2(b)$. Clearly, $\mathcal{G}_{\text{lgg}} \equiv_{\mathcal{R}} \mathcal{G}'_{\text{lgg}}$ holds. Let us show that $\mathcal{G} \models_{\mathcal{R}} \mathcal{G}'_{\text{lgg}}$ holds, i.e., there exists a homomorphism ϕ from the blank nodes in $\mathcal{G}'_{\text{lgg}}$ to the terms in \mathcal{G}^∞ such that $[\mathcal{G}'_{\text{lgg}}]_{\phi} \subseteq \mathcal{G}^\infty$. By construction of \mathcal{G} , it is easy to see that the above holds when ϕ maps $b_{v_1 v_2}$ 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 $\mathcal{G} \models_{\mathcal{R}} \mathcal{G}'_{\text{lgg}}$, hence $\mathcal{G} \models_{\mathcal{R}} \mathcal{G}_{\text{lgg}}$. \square

As an immediate consequence of the above results, we get the following worst-case bounds for the time to compute a cover graph-based **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^∞ and \mathcal{G}_2^∞ , as the times to compute them and their sizes depend on the 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 $|\mathcal{G}_1^\infty| \times |\mathcal{G}_2^\infty|$.*

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

Figure 2.2 (bottom) displays a cover graph-based **lgg** of the RDF graphs \mathcal{G}_1 and \mathcal{G}_2 in Figure 2.1 w.r.t. the entailment rules shown in Table 1.2. In contrast to Figure 2.2 (top), 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 gray ones) yields precisely the $\mathbf{l}gg$ $\mathcal{G}_{\mathbf{l}gg}$ of \mathcal{G}_1 and \mathcal{G}_2 shown in Figure 2.1.

2.4 Conclusion

In this chapter, we first defined the notion of an $\mathbf{l}gg$ of RDF graphs (Definition 1) faithful to the entire W3C standard, for which we showed (Theorem 1) that there exists a unique equivalent class of $\mathbf{l}ggs$ of RDF graphs: though syntactically different $\mathbf{l}ggs$ of some RDF graphs exist, they are all equivalent w.r.t. the standard relation of entailment between RDF graphs. We also showed that the problem (Problem 1) of computing such an $\mathbf{l}gg$ of $n > 2$ RDF graphs can be solved, without loss of generality, if we have a solution to the particular instance of this problem where $n = 2$ (Proposition 1).

We therefore devised the notion of cover graph of two input RDF graphs (Definition 3), which is an RDF graph, for which we showed that it corresponds (*i*) to an $\mathbf{l}gg$ of the input RDF graphs when no entailment rules are considered (Theorem 2), i.e., when the semantics of RDF graph based on RDF entailment is ignored, and to (*ii*) an $\mathbf{l}gg$ of the input RDF graphs w.r.t. the entailment rules under consideration (Theorem 3), when the input RDF graphs are saturated with these rule before computing their cover graph. Further, we showed (Proposition 2 and Corollary 1) that the time to compute a cover graph, and its size, are at most quadratic in the size of the input (saturated) RDF graphs.

Chapter 3

Finding Commonalities between SPARQL queries

Contents

3.1	Introduction	37
3.2	Comparing Queries w.r.t. Ontological Constraints	39
3.3	Learning the lgg of Queries w.r.t. Ontological Constraints	42
3.4	Computing an lgg of Queries w.r.t. ontological constraints	44
3.5	Conclusion	49

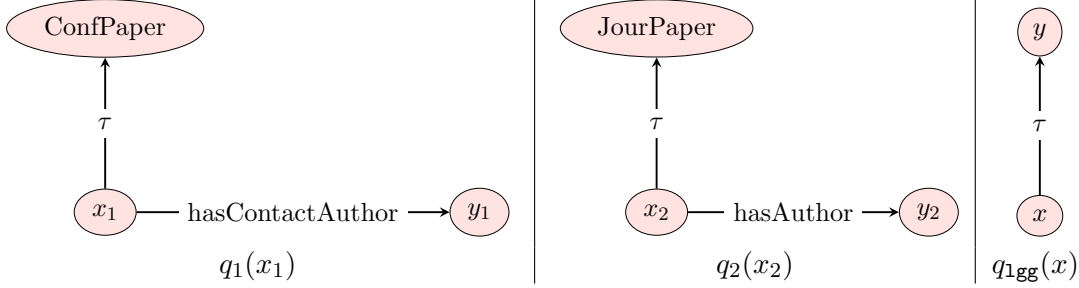
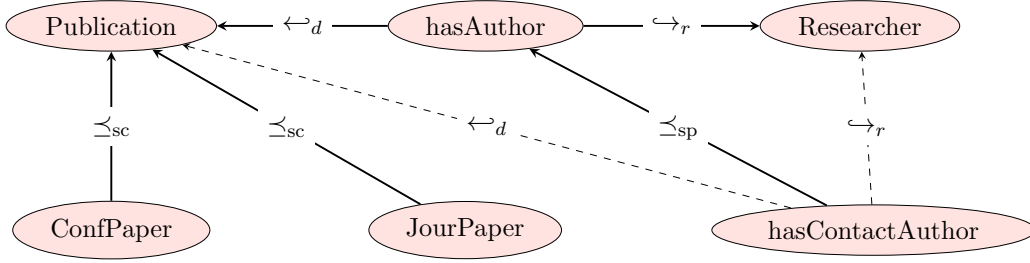
3.1 Introduction

In this chapter, we present the second main contribution of this thesis: we define and solve the problem of learning the commonalities between conjunctive SPARQL queries, i.e., BGPQs, formalized as an **lgg** of these queries w.r.t. a well-founded extension of the standard relation of entailment between BGPQs. Indeed, one may be tempted to adapt to our SPARQL setting the notion of **lgg** for RDF graphs (Definition 1) by using *BGPQs* in place of RDF graphs and the off-the-shelf *entailment between BGPQs* instead of entailment between RDF graphs:

Definition 4 (**lgg** of BGPQs). *Let q_1, \dots, q_n be BGPQs with the same arity and \mathcal{R} a set of RDF entailment rules.*

- *A generalization of q_1, \dots, 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, \dots, q_n is a generalization q_{lgg} of q_1, \dots, q_n such that for any other generalization q_g of q_1, \dots, q_n : $q_{\text{lgg}} \models_{\mathcal{R}} q_g$.*

Unfortunately, this straightforward definition is of limited practical interest as the next example shows. Consider the BGPQs q_1 and q_2 in Figure 3.1, 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 1.2, an

Figure 3.1 – Sample BGPQs q_1, q_2 and their minimal lgg q_{lgg} .Figure 3.2 – Sample set \mathcal{O} of RDFS ontological constraints .

lgg of q_1 and q_2 is the *very* general BGPQ $q_{\text{lgg}}(x) \leftarrow (x, \tau, y)$ asking for *the resources having some type* shown in Figure 3.1 (right).

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 Figure 3.2 that hold in the scientific publication domain, a more precise lgg for the above-mentioned q_1, q_2 would be $q_{\text{lgg}}(x) \leftarrow (x, \tau, \text{Publication}), (x, \text{hasAuthor}, y), (y, \tau, \text{Researcher})$ 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. It is worth noting that, while background knowledge is part of an RDF graph when it contains RDFS constraints, background knowledge is *not* part of a BGPQ even if it contains RDFS constraints: a BGPQ only defines select-project-join search conditions to be matched against an RDF graph.

To define such more precise lgg s and state our learning problem in Section 3.3, we start by generalizing the standard specialization/generalization relation of *entailment between BGPQs* in Section 3.2, 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. Finally, we present our solution to the

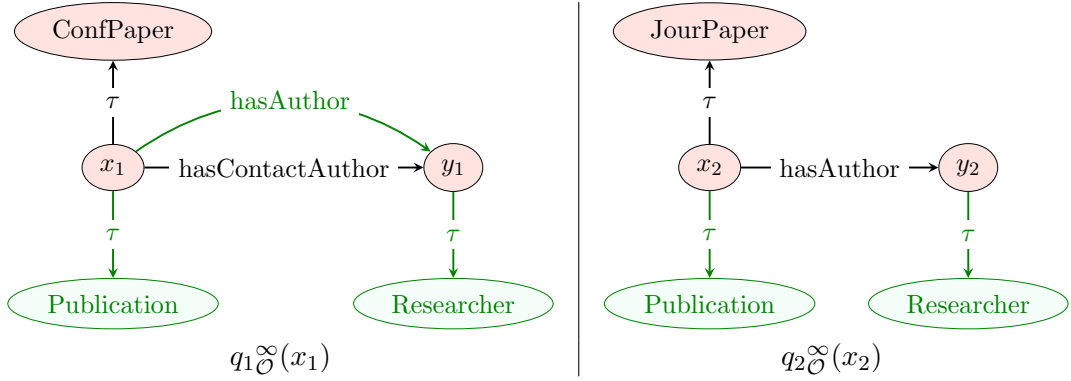


Figure 3.3 – Saturations of the BGPQs q_1 and q_2 from Figure 3.1 w.r.t. \mathcal{O} from Figure 3.2. Triples shown in green are added by saturation.

novel learning problem we identified in Section 3.4.

3.2 Comparing Queries w.r.t. Ontological Constraints

Our new entailment relation between queries builds on the following notion of *saturation of a query*, which materializes the semantics of the query in the light of some given background knowledge:

Definition 5 (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 $q_\mathcal{O}^\infty$, is the BGPQ with the same answer variables as q and whose body, noted $\text{body}(q_\mathcal{O}^\infty)$, is the maximal subset of $(\text{body}(q) \cup \mathcal{O})^\infty$ such that for any of its subset \mathcal{S} : if $\mathcal{O} \models_{\mathcal{R}} \mathcal{S}$ holds then $\text{body}(q) \models_{\mathcal{R}} \mathcal{S}$ holds.*

Observe that this saturation *coincides* with that of BGPQs (Section 1.3) when the set of RDF constraints \mathcal{O} is empty. Also, such a saturation is pertinent just in case the RDF entailment rules under consideration utilize the RDFS constraints, e.g., those in Table 1.2; otherwise the set of constraints is useless.

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 $(\text{body}(q) \cup \mathcal{O})^\infty$ shown in Figure 3.4:

$$\text{body}(q_\mathcal{O}^\infty) = (\text{body}(q) \cup \mathcal{O})^\infty \setminus (\mathcal{O}^\infty \setminus \text{body}(q)^\infty).$$

Figure 3.3 illustrates the above notion of saturation using the BGPQs and RDFS constraints from Figure 3.2.

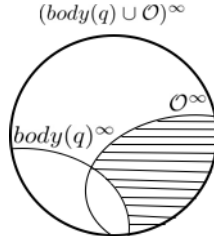


Figure 3.4 – Characterization of the body of a saturated BGPQ q w.r.t. a set \mathcal{O} of RDFS constraints

The next theorem establishes that our novel saturation is *well-founded* w.r.t. the central reasoning tasks of query entailment and query answering, i.e., a BGPQ and its saturation are equivalent w.r.t. the query entailment and query answering viewpoints:

Theorem 4. *Let \mathcal{R} be a set of RDF entailment rules, \mathcal{O} a set of RDFS statements, and q a BGPQ whose saturation w.r.t. \mathcal{O} is $q_{\mathcal{O}}^\infty$. 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. $q(\mathcal{G}) = q_{\mathcal{O}}^\infty(\mathcal{G})$ holds.

Proof. Let us first show item 1.

Showing that $\mathcal{G} \models_{\mathcal{R}} q$ iff $\mathcal{G} \models_{\mathcal{R}} q_{\mathcal{O}}^\infty$ holds amounts to showing that $\mathcal{G}^\infty \models q$ iff $\mathcal{G}^\infty \models q_{\mathcal{O}}^\infty$ holds (recall Chapter 1).

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 $\mathcal{G}^\infty \models q$ holds then $\mathcal{G}^\infty \models body(q) \cup \mathcal{O}$ holds since $\mathcal{O} \subseteq \mathcal{G}$, hence $\mathcal{G}^\infty \models (body(q) \cup \mathcal{O})^\infty$ holds (because $\mathcal{G}^\infty \models body(q) \cup \mathcal{O}$ holds iff there exists a homomorphism ϕ such that $[body(q) \cup \mathcal{O}]_\phi \subseteq \mathcal{G}^\infty$ and therefore $[body(q) \cup \mathcal{O}]_\phi^\infty \subseteq \mathcal{G}^\infty$ holds). By definition of the saturation of a BGPQ, $body(q_{\mathcal{O}}^\infty) \subseteq (body(q) \cup \mathcal{O})^\infty$, thus $\mathcal{G}^\infty \models body(q_{\mathcal{O}}^\infty)$ holds.

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. \square

Finally, the next theorem states the computational complexity of our novel saturation of a query:

Theorem 5. *The saturation of a BGPQ w.r.t. a set of RDFS constraints can be computed in polynomial time.*

Proof. The polynomial time result directly follows from the fact that, in Definition 5, (i) $q_{\mathcal{O}}^\infty$ has the same head variables as q (hence these can be computed in linear time) and (ii) the body of $q_{\mathcal{O}}^\infty$ can be computed based on the set characterization displayed in Figure 3.4, i.e., as $(body(q) \cup \mathcal{O})^\infty \setminus (\mathcal{O}^\infty \setminus body(q)^\infty)$ (hence in polynomial time, since the saturation of BGP/RDF graphs can be computed in polynomial time (Chapter 1)). \square

With the above notion of saturation in place, we endow entailment between queries with background knowledge as follows:

Definition 6 (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 $q_{\mathcal{O}}^{\infty} \models q'$ holds.*

Clearly, the above definition *coincides* with standard RDF entailment between BGPQs when \mathcal{O} is empty (recall Section 1.3).

Using the set \mathcal{R} of entailment rules in Table 1.2, the above mentioned BGPQ $q_{1\text{gg}}(x) \leftarrow (x, \tau, \text{Publication}), (x, \text{hasAuthor}, y), (y, \tau, \text{Researcher})$ is *neither* entailed by q_1 *nor* by q_2 from Figure 3.1, 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 Figure 3.3: $q_{1\mathcal{O}}^{\infty} \models^{\phi_1} q$ holds for $\phi_1(x) = x_1$ and $\phi_1(y) = y_1$, and $q_{2\mathcal{O}}^{\infty} \models^{\phi_2} q$ holds for $\phi_2(x) = x_2$ and $\phi_2(y) = y_2$.

Further, the main theorem below states that our novel entailment relation is *well-founded* w.r.t. the central reasoning tasks of query entailment and query answering, i.e., when a query is entailed by another w.r.t. some background knowledge, similarly to Property 4 in Section 1.3 where no background is considered, the former generalizes the latter from the query entailment and query answering viewpoints:

Theorem 6. *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. *$q(\mathcal{G}) \subseteq q'(\mathcal{G})$ holds.*

Proof. Let us first show item 1.

By Theorem 4, item 1., $\mathcal{G} \models_{\mathcal{R}} q$ holds iff $\mathcal{G} \models_{\mathcal{R}} q_{\mathcal{O}}^{\infty}$ holds. Consider some ϕ such that $\mathcal{G} \models_{\mathcal{R}}^{\phi} q_{\mathcal{O}}^{\infty}$ holds. Also, by Definition 6, consider some ϕ' such that $q_{\mathcal{O}}^{\infty} \models^{\phi'} q'$. By composing ϕ and ϕ' into ψ , we get $\mathcal{G} \models_{\mathcal{R}}^{\psi} q'$, 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 $\mathcal{G} \models_{\mathcal{R}}^{\phi} q_{\mathcal{O}}^{\infty}$ and $q_{\mathcal{O}}^{\infty} \models^{\phi'} q'$ hold, and (ii) q and $q_{\mathcal{O}}^{\infty}$ have, by definition, the same answer variables. \square

Finally, we establish that entailment between queries w.r.t. background knowledge is as hard as standard entailment between queries:

Theorem 7. *Deciding entailment between two BGPQs w.r.t. RDFS constraints is NP-complete.*

Proof. The problem is NP-hard because entailment between two BGPQs w.r.t. RDFS constraints collapses to standard entailment between BGPQs, which is NP-complete (recall Section 1.3), when the set of RDFS constraints under consideration is empty.

The problem is in NP since, as stated in Definition 6, $q \models_{\mathcal{R}, \mathcal{O}} q'$ iff $q_{\mathcal{O}}^{\infty} \models q'$, i.e., $q \models_{\mathcal{R}, \mathcal{O}} q'$ can be decided by first computing $q_{\mathcal{O}}^{\infty}$ in polynomial time (Theorem 5) and then using simple entailment between BGPQs, which is NP-complete (recall Section 1.3), to check if $q_{\mathcal{O}}^{\infty} \models q'$ holds. \square

3.3 Learning the lgg of Queries w.r.t. Ontological Constraints

In the light of the preceding results, we revise/generalize Definition 4 as follows:

Definition 7 (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, \dots, q_n BGPQs with the same arity.*

- A generalization of q_1, \dots, 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, \dots, q_n w.r.t. \mathcal{O} is a generalization q_{lgg} of q_1, \dots, q_n w.r.t. \mathcal{O} such that for any other generalization q_g of q_1, \dots, q_n w.r.t. \mathcal{O} : $q_{\text{lgg}} \models_{\mathcal{R}, \mathcal{O}} q_g$.

By contrast with an lgg of RDF graphs that always exists (Theorem 1), we found:

Theorem 8. *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 ($\models_{\mathcal{R}, \mathcal{O}}$).*

Proof. It is easy to show that an lgg of BGPQs may not exist. For instance, consider the BGPQs $q_1(x_1) \leftarrow (x_1, \text{hasAuthor}, y_1)$ asking for the resources having some author, and $q_2(x_2) \leftarrow (y_2, \text{hasAuthor}, x_2)$ asking for the authors of some resource. Clearly, when the set \mathcal{R} of entailment rules is empty or comprises the rules in Table 1.2, 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 $q_{\text{lgg}}(x) \leftarrow (x, \tau, \text{rdf:Resource})$, since every RDF value is an instance of the built-in class `rdf:Resource`.

When an lgg of BGPQs w.r.t. RDFS constraints exists, it is unique up to entailment, i.e., is semantically unique, because $q_{\text{lgg}} \models_{\mathcal{R}, \mathcal{O}} q_g$ holds for any q_g in Definition 4. If it were that queries have *multiple lgg's incomparable* w.r.t. entailment, say the BGPQs $\text{lgg}_1(\bar{x}), \dots, \text{lgg}_m(\bar{x})$, the BGPQ defined as $q_{\text{lgg}}(\bar{x}) \leftarrow \text{body}(\text{lgg}_1) \uplus \dots \uplus \text{body}(\text{lgg}_m)$ ¹ would be a *single strictly more specific lgg*, a contradiction. \square

1. \uplus generalizes the merge operator for RDF graphs. It corresponds to the union of the input BGPQ bodies after renaming their non-answer variables with fresh ones.

Similarly to the case of RDF graphs, though unique up to entailment, there exist many syntactic variants (an infinity actually) of an lgg of BGPQs due to *redundant* triples, i.e., triples entailed by others within the lgg . For example, think of an lgg $q_{\text{lgg}}(x) \leftarrow (x, \tau, A), (x, \tau, B), (x, y, z)$ w.r.t. the set of constraints $\mathcal{O} = \{(A, \preceq_{\text{sc}}, B), (B, \preceq_{\text{sc}}, A)\}$, 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 $q_{\text{lgg}}(x) \leftarrow (x, \tau, A)$ and $q_{\text{lgg}}(x) \leftarrow (x, \tau, B)$, since (x, y, z) is entailed by each of the two other triples, and (x, τ, B) is entailed by (x, τ, A) w.r.t. \mathcal{O} , and vice versa, because A and B are equivalent.

Again, importantly, redundancy of triples is not specific to lggs of BGPQs w.r.t. RDFS constraints, since obviously any BGPQ may feature redundancy. The detection and elimination of such redundancy have been studied in the literature [Meier, 2008, Pichler et al., 2013], hence we focus in this work on learning *some* lgg of BGPQs w.r.t. RDFS constraints; learning as minimal as possible lggs is a perspective of this work discussed in Section 7.2.

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

Problem 2. *Given a set \mathcal{R} of RDF entailment rules, a set \mathcal{O} of RDFS statements, and the BGPQs q_1, \dots, q_n with the same arity, find an lgg of q_1, \dots, q_n w.r.t. \mathcal{O} .*

The proposition below, which is the counterpart of Proposition 1 that we established for RDF graphs, shows that an lgg of $n \geq 3$ BGPQs can be inductively 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:

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

Proposition 3. *Let $q_1, \dots, q_{n \geq 3}$ be n BGPQs, \mathcal{O} a set of RDFS statements and \mathcal{R} a set of RDF entailment rules. q_{lgg} is an lgg of q_1, \dots, q_n w.r.t. \mathcal{O} iff q_{lgg} is an lgg w.r.t. \mathcal{O} of an lgg of q_1, \dots, q_{n-1} w.r.t. \mathcal{O} and q_n .*

Proof. The proof relies on the next lemma.

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

Let us show that the above lemma holds.

Suppose that q_{lgg}^1 is an lgg of $q_1, \dots, q_{k < n}$ w.r.t. \mathcal{O} , i.e., by Definition 4: (i) $q_{1 \leq i \leq k} \models_{\mathcal{R}, \mathcal{O}} q_{\text{lgg}}^1$ holds and (ii) for any BGPQ q such that $q_{1 \leq i \leq k} \models_{\mathcal{R}, \mathcal{O}} q$ holds, $q_{\text{lgg}}^1 \models_{\mathcal{R}, \mathcal{O}} q$ holds.

Suppose also that $q_{1\text{gg}}^2$ is an **lgg** of q_1, \dots, q_n w.r.t. \mathcal{O} , i.e., by Definition 4: (i) $q_{1 \leq i \leq n} \models_{\mathcal{R}, \mathcal{O}} q_{1\text{gg}}^2$ holds and (ii) for any BGPQ q' such that $q_{1 \leq i \leq n} \models_{\mathcal{R}, \mathcal{O}} q'$ holds, $q_{1\text{gg}}^2 \models_{\mathcal{R}, \mathcal{O}} q'$ holds.

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

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

(\Rightarrow) Assume that $q_{1\text{gg}}$ is an **lgg** of q_1, \dots, 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, \dots, q_{n-1} and q_n w.r.t. \mathcal{O} . By Definition 4, because $q_{1\text{gg}}$ is an **lgg** of q_1, \dots, q_n w.r.t. \mathcal{O} , we have: (i) $q_{1 \leq i \leq n} \models_{\mathcal{R}, \mathcal{O}} q_{1\text{gg}}$ holds and (ii) for any BGPQ q such that $q_{1 \leq i \leq n} \models_{\mathcal{R}, \mathcal{O}} q$ holds, $q_{1\text{gg}} \models_{\mathcal{R}, \mathcal{O}} q$ holds.

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

From Definition 4, (*) and (**) we get that $q_{1\text{gg}}$ is an **lgg** w.r.t. \mathcal{O} of an **lgg** of q_1, \dots, q_{n-1} and q_n w.r.t. \mathcal{O} .

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

By Definition 4, (i) $q' \models_{\mathcal{R}, \mathcal{O}} q_{1\text{gg}}$ and $q_n \models_{\mathcal{R}, \mathcal{O}} q_{1\text{gg}}$ hold, hence $q_{1 \leq i \leq n-1} \models_{\mathcal{R}, \mathcal{O}} q_{1\text{gg}}$ and $q_n \models_{\mathcal{R}, \mathcal{O}} q_{1\text{gg}}$ hold, i.e., (*) $q_{1 \leq i \leq n} \models_{\mathcal{R}, \mathcal{O}} q_{1\text{gg}}$ 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, $q_{1\text{gg}} \models_{\mathcal{R}, \mathcal{O}} q$ holds. By Definition 4, $q_{1 \leq i \leq n-1} \models_{\mathcal{R}, \mathcal{O}} q'$ holds, therefore we get: (**') for any BGPQ q such that $q_{1 \leq i \leq n} \models_{\mathcal{R}, \mathcal{O}} q$ holds, $q_{1\text{gg}} \models_{\mathcal{R}, \mathcal{O}} q$ holds.

From Definition 4, (*) and (**') we get that $q_{1\text{gg}}$ is an **lgg** of q_1, \dots, q_n w.r.t. \mathcal{O} . \square

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

3.4 Computing an lgg of Queries w.r.t. ontological constraints

Our solution to the above learning problem (Problem 2) adapts that we devised for RDF graphs to BGPQs. We first define the notion of *anti-unification* of BGPQ triple patterns (Definition 8), based on which we devise the notion of *cover query* of two BGPQs (Definition 9). Then, we establish that the cover query of two BGPQs q_1 and q_2 is an **lgg** of q_1 and q_2 *just in case* both RDF entailment rules and ontological constraints are ignored (Theorem 9). Further, we show (Theorem 10) that an **lgg** of q_1 and q_2 as defined in Definition 7, 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 5). Importantly, we also show that the existence of such cover queries and **lggs** coincide.

We also provide the size of these cover query-based lgg (i.e., number of triples), as well as the time to compute them.

We transfer the notion of anti-unification [Plotkin, 1970, Plotkin, 1971, Robinson and Voronkov, 2001] to the SPARQL setting as follows:

Definition 8 (Anti-unification of triple patterns). *Let ς be an injective generalization function from $(\mathcal{U} \cup \mathcal{L} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{L} \cup \mathcal{V})$ to $\mathcal{U} \cup \mathcal{L} \cup \mathcal{V}$ that maps (i) any pair of same input URI or literal value to itself, i.e., $\varsigma(v, v) = v$, and (ii) any other pair of values (URIs, literals, variables and mix thereof) to a variable.*

The anti-unification of the two triples (t_1, t_2, t_3) and (t_4, t_5, t_6) is the triple $(\varsigma(t_1, t_4), \varsigma(t_2, t_5), \varsigma(t_3, t_6))$.

Observe that, by contrast with RDF graph triples, there is no restriction on the BGPQ triples that can be anti-unified, as any triples from $(\mathcal{U} \cup \mathcal{L} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{L} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{L} \cup \mathcal{V})$ is *well-formed*.

We are now able to define our central notion of cover query:

Definition 9 (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) = $q(x_1^1, \dots, x_1^n)$ and head(q_2) = $q(x_2^1, \dots, x_2^n)$ iff head(q) = $q(v_{x_1^1 x_2^1}, \dots, v_{x_1^n x_2^n})$*
- *$(t_1, t_2, t_3) \in \text{body}(q_1)$ and $(t_4, t_5, t_6) \in \text{body}(q_2)$ iff $(t_7, t_8, t_9) \in \text{body}(q)$ 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 .

It is worth noting that the definition of cover query, like that of cover graph (Definition 3), graph does not manipulate explicitly the generalization function ς . It is left implicitly in our way of defining the value of t_i for $1 \leq i \leq 3$: $t_{i+6} = t_i$ if $t_i = t_{i+3}$ and $t_i \in \mathcal{U} \cup \mathcal{L}$ corresponds to the particular case $\varsigma(v, v) = v$ for v a URI or literal, while *otherwise t_{i+6} is the variable $v_{t_i t_{i+3}}$* corresponds to all the other cases, i.e., $t_{i+6} = v_{t_i t_{i+3}} = \varsigma(t_i, t_{i+3})$.

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), trivially extended to the fact these queries can be of any arity, 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 7 with $\mathcal{R} = \emptyset$ and $\mathcal{O} = \emptyset$). Moreover, similarly to the blank nodes introduced in the cover graph of RDF graphs (Section 2.3), (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

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 9, 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 Section 3.3 to point out that an lgg may no exist. Their cover query does not exist either, because Definition 9 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 9. *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 10 (see below), noting that in the particular case of Theorem 9 $\mathcal{R} = \emptyset$ and $\mathcal{O} = \emptyset$ hold, hence $q_1^\infty = q_1$ and $q_2^\infty = q_2$ hold. \square

It follows from the above result that the cover query q of two BGPQs q_1 and q_2 displayed in Figure 3.5 (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; they directly follow from the definition of cover query (Definition 9).

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

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

Theorem 10. *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^∞, q_2^∞ exists iff an lgg of q_1, q_2 w.r.t. \mathcal{O} exists;*
2. *the cover query q of q_1^∞, q_2^∞ is an lgg of q_1, 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 the 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 $v_{t_1 t_2}$ by the value t_1 . Clearly, $q' \models_{\mathcal{R}, \mathcal{O}} q$ and, by construction of q , $q' = q_1^\infty$, 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 $q_{1\text{lgg}}$ 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, Chapter 1). Also, assume that $q, q_1, q_2, q_{1\text{lgg}}$ heads are $q(\bar{x}), q_1(\bar{x}_1), q_2(\bar{x}_2), q_{1\text{lgg}}(\bar{x}_{1\text{lgg}})$ respectively.

To prove our claim, we need to show that $q \models_{\mathcal{R}, \mathcal{O}} q_{1\text{lgg}}$ holds. Since $q_{1\text{lgg}}$ 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 $q_{1\text{lgg}}$ to the values (variables and constants) in q_1^∞ and in q_2^∞ respectively, such that $[body(q_{1\text{lgg}})]_{\phi_1} \subseteq body(q_1^\infty)$ and $\phi_1(\bar{x}_{1\text{lgg}}) = \bar{x}_1$, and similarly, $[body(q_{1\text{lgg}})]_{\phi_2} \subseteq body(q_2^\infty)$ and $\phi_2(\bar{x}_{1\text{lgg}}) = \bar{x}_2$.

Consider the BGPQ $q'_{1\text{lgg}}$ obtained from $q_{1\text{lgg}}$ by replacing every variable v by $v_{t_1 t_2}$ where $t_1 = \phi_1(v)$ and $t_2 = \phi_2(v)$. Clearly, $q_{1\text{lgg}} \equiv_{\mathcal{R}, \mathcal{O}} q'_{1\text{lgg}}$ holds. Assume that the head of $q'_{1\text{lgg}}$ is $q'_{1\text{lgg}}(\bar{x}'_{1\text{lgg}})$; by construction it has the same head as q .

Let us show that $q \models_{\mathcal{R}, \mathcal{O}} q'_{1\text{lgg}}$ holds, i.e., there exists a homomorphism ϕ from the variables in $q'_{1\text{lgg}}$ to values in q^∞ such that $[body(q'_{1\text{lgg}})]_\phi \subseteq body(q^\infty)$ and $\phi(\bar{x}'_{1\text{lgg}}) = \bar{x}$.

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. Crucially, this entails (i) the existence of the cover query q when an **lgg** $q_{1\text{lgg}}$ exists and (ii) that $q \models_{\mathcal{R}, \mathcal{O}} q'_{1\text{lgg}}$ holds, hence $q \models_{\mathcal{R}, \mathcal{O}} q_{1\text{lgg}}$ holds, and therefore q is an **lgg** of q_1 and q_2 w.r.t. \mathcal{O} . \square

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^∞ and q_2^∞ 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^∞ and q_2^∞ , and their sizes, depend on the particular sets \mathcal{O} and \mathcal{R} at hand.

Corollary 2. *A cover query-based **lgg** of two BGPQs q_1 and q_2 is computed in $\mathcal{O}(|body(q_1^\infty)| \times |body(q_2^\infty)|)$ and its size is $|body(q_1^\infty)| \times |body(q_2^\infty)|$.*

Figure 3.5 (bottom) displays the cover query of the BGPQs q_1^∞ and q_2^∞ shown in Figure 3.3. It is therefore (Theorem 10) an **lgg** of the BGPQs q_1 and q_2 , shown in Figure 3.1 w.r.t. the set \mathcal{O} of RDFS constraints, shown in Figure 3.2, using the RDF entailment rules shown in Table 1.2.

Figure 3.5 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*.

3.5 Conclusion

In this chapter, we first devised an extension of the standard relation of entailment between BGPQs (Definition 6), in order to compare them w.r.t. background knowledge formalized as RDFS ontological constraints. In particular, we showed that this novel generalization/specialization relation is well-founded w.r.t. the central reasoning tasks of query entailment and of query answering (Theorem 6) and that deciding entailment between BGPQs w.r.t. RDFS constraints is NP-complete (Theorem 7). Crucially, the definition of this relation builds on a notion BGPQ saturation w.r.t. background knowledge (Definition 5), which leverages the relevant background knowledge to complement the query.

Then, we adapted the notion of $\mathbf{1gg}$ that we defined for RDF graphs in the preceding chapter, to our SPARQL setting by using BGPQs in place of RDF graphs and our new entailment relation between BGPQs in place of entailment between RDF graphs (Definition 7), because we pointed out that the shortcomings of using standard entailment between BGPQs. Notably, and by contrast with the notion of $\mathbf{1gg}$ of RDF graphs, we showed that an $\mathbf{1gg}$ of BGPQs may not exist (Theorem 8), whether or not we consider background knowledge. Similarly to the notion of $\mathbf{1gg}$ of RDF graphs, we showed (Theorem 8) that - when an $\mathbf{1gg}$ exists - there exists a unique equivalent class of $\mathbf{1ggs}$ of BGPQs: though syntactically different $\mathbf{1ggs}$ of some BGPQs exist, they are all equivalent w.r.t. the relation of entailment between BGPQs. We also showed that the problem (Problem 2) of computing such an $\mathbf{1gg}$ of $n > 2$ BGPQs can be solved, without loss of generality, if we have a solution to the particular instance of this problem where $n = 2$ (Proposition 3).

We therefore devised the notion of cover query of two input BGPQs (Definition 9), which is a BGPQ, whose existence coincides with that of their $\mathbf{1gg}$ and, further, this cover query corresponds (i) to an $\mathbf{1gg}$ of the input BGPQs when no entailment rules and background knowledge are considered (Theorem 9) and to (ii) an $\mathbf{1gg}$ of the input BGPQs w.r.t. the entailment rules and background knowledge under consideration (Theorem 10), when the input BGPQs are saturated with these rules and knowledge before computing their cover query. Also, we showed (Proposition 4 and Corollary 2) that the time to compute a cover query, and its size, are at most quadratic in the size of the input (saturated) BGPQs.

Chapter 4

Algorithms

Contents

4.1	Introduction	51
4.2	Least general anti-unification	51
4.3	lggs of RDF graphs	52
4.3.1	Handling large RDF graphs using DMSs	53
4.3.2	Handling huge RDF graphs using MapReduce	55
4.4	lggs of BGPQs	56
4.5	Conclusion	57

4.1 Introduction

In this chapter we provide algorithms to compute lggs of RDF graphs and BGPQs, based on the results obtained in the two preceding chapters.

In Section 4.2, we present an algorithm for computing the least general anti-unifiers of tuples of RDF values (URIs, literals, blank nodes and variables), i.e., of triples, triple patterns or query heads, on which the definitions of cover graphs and queries rely. Then, in Section 4.3, we give three algorithms to compute a cover graph-based lgg of RDF graphs, which allow handling RDF graphs of increasing size, i.e., when the input and output RDF graphs fit in memory, in data management systems or in MapReduce clusters. Also, to choose between these algorithms, we show how the exact size of a cover graph-based lgg they produce can be calculated, *without computing this lgg*. Finally, in Section 4.4, we provide an in-memory algorithm to compute a cover query-based lgg of BGPQs.

4.2 Least general anti-unification

Algorithm 1, called `lgau`, computes a least general anti-unifier (t'_1, \dots, t'_n) of two n -ary tuples of RDF values (t_1, \dots, t_n) and (t'_1, \dots, t'_n) , made of constants (URIs and

Algorithm 1 Least general anti-unification: `lgau`**In:** $T_1 = (t_1, \dots, t_n)$ and $T_2 = (t'_1, \dots, t'_n)$, boolean *bnodes***Out:** least general anti-unification T of T_1 and T_2

```

1: for  $i = 1$  to  $n$  do                                ▷ for each pair of  $i$ th attributes
2:   if  $t_i = t'_i$  and  $t_i \in \mathcal{U} \cup \mathcal{L}$  then
3:      $t_i^T \leftarrow t_i$                                 ▷ generalization of a constant by itself
4:   else if bnodes then                                ▷ cover graph case
5:      $t_i^T \leftarrow b_{t_i, t'_i}$                        ▷ generalization by a blank node
6:   else                                                ▷ cover query case
7:      $t_i^T \leftarrow v_{t_i, t'_i}$                        ▷ generalization by a variable
8: return  $(t_1^T, \dots, t_n^T)$                           ▷ () when  $n = 0$ 

```

literals), blank nodes and variables. This is achieved by setting the i^{th} value t_i^T of the output tuple to the least general generalization of the values found at the i^{th} positions of the two input tuples: t_i and t'_i . Recall that a pair of a same constant is generalized by that constant itself, while in all other cases the generalization is either a blank node in case of RDF graph triple values (Section 2.3) or a variable in case of query head answer variables or of query body triple pattern values (Section 3.4). Generalizing pairs of different values with blank nodes or variables is controlled with the Boolean *bnodes* parameter. Crucially, these generated blank nodes (resp. variables) adopt the consistent naming scheme devised in Section 2.3 (resp. in Section 3.4) that allows us preserving the common input RDF graphs (resp. BGPQs) structure across the anti-unifications of triples (resp. triple patterns).

4.3 lggs of RDF graphs

Following Definition 3, Algorithm 2, called `lgg4g`, computes the cover graph \mathcal{G} of two input RDF graphs \mathcal{G}_1 and \mathcal{G}_2 : \mathcal{G} comprises the least general anti-unification of every pair of \mathcal{G}_1 and \mathcal{G}_2 triples with *same* property. Therefore, given two RDF graphs \mathcal{G}_1 and \mathcal{G}_2 , a call `lgg4g($\mathcal{G}_1, \mathcal{G}_2$)` produces the cover graph-based `lgg` of \mathcal{G}_1 and \mathcal{G}_2 ignoring RDF entailment (Theorem 2), while a call `lgg4g($\mathcal{G}_1^\infty, \mathcal{G}_2^\infty$)` produces the cover graph-based `lgg` of \mathcal{G}_1 and \mathcal{G}_2 taking into account the set of RDF entailment rules at hand (Theorem 3). In the latter case, the input RDF graphs can be saturated using standard algorithms implemented in RDF data management systems, like Jena [jen, 2017] and Virtuoso [vir, 2017].

Importantly, `lgg4g` assumes that the input RDF graphs, as well as their output cover graph, fit in memory. Checking whether this is the case for the input RDF graphs under consideration can be done as follows.

The size of the input RDF graphs \mathcal{G}_1 and \mathcal{G}_2 can be computed with the following SPARQL queries counting how many triples each of them holds:

Algorithm 2 Cover graph of two RDF graphs: **lgg4g**

In: RDF graphs \mathcal{G}_1 and \mathcal{G}_2 **Out:** \mathcal{G} is the cover graph of \mathcal{G}_1 and \mathcal{G}_2

- 1: $\mathcal{G} \leftarrow \emptyset$
 - 2: **for all** $T_1 = (s_1, p_1, o_1) \in \mathcal{G}_1$ **do**
 - 3: **for all** $T_2 = (s_2, p_2, o_2) \in \mathcal{G}_2$ with $p_1 = p_2$ **do**
 - 4: $\mathcal{G} \leftarrow \mathcal{G} \cup \{\text{lgau}(T_1, T_2, \text{true})\}$
 - 5: **return** \mathcal{G}
-

SELECT count(*) as ?size FROM \mathcal{G}_i with $i \in [1, 2]$.

Recall that the worst-case size of the output cover graph is $|\mathcal{G}| = |\mathcal{G}_1| \times |\mathcal{G}_2|$ in the unlikely case where *all* the \mathcal{G}_1 and \mathcal{G}_2 triples use the same property (Property 2 and Corollary 1).

The precise size of the output cover graph \mathcal{G} can be computed, *without* computing \mathcal{G} , with SPARQL queries. First, we calculate for each input RDF graph \mathcal{G}_i , with $i \in [1, 2]$, how many triples it holds per distinct property p :

$$S_{\mathcal{G}_i} = \{(p, n_i) \mid |\{(s, p, o) \in \mathcal{G}_i\}| = n_i\}$$

This can be computed with the SPARQL query:

SELECT ?p count(*) as ?n_i FROM \mathcal{G}_i WHERE $\{(?s, ?p, ?o)\}$ GROUP BY ?p.

Then, since every \mathcal{G}_1 triple with property p anti-unifies with every \mathcal{G}_2 triple with same property p , in order to beget \mathcal{G} , the size of \mathcal{G} is:

$$|\mathcal{G}| = \sum_{(p, n_1) \in S_{\mathcal{G}_1}, (p, n_2) \in S_{\mathcal{G}_2}} n_1 \times n_2$$

This can be computed with the SPARQL query:

SELECT SUM(?n₁*?n₂) as ?size WHERE $\{\{S_{\mathcal{G}_1}\}\{S_{\mathcal{G}_2}\}\}$

with $S_{\mathcal{G}_1}$ and $S_{\mathcal{G}_2}$ denoting the above-mentioned SPARQL queries computing these two sets, which join on their common answer variable $?p$.

When the input RDF graphs or their output cover graph cannot fit in memory, we propose variants of **lgg4g** that either assume that RDF graphs are stored in data management systems (DMSs, in short) or in a MapReduce cluster.

4.3.1 Handling large RDF graphs using DMSs

Algorithm 3, called **lgg4g-dms**, is an adaptation of **lgg4g**, which assumes that the input RDF graphs (already saturated if needed) and their cover graph are all

Algorithm 3 Cover graph of two RDF graphs: `lgg4g-dms`

In: cursor c_1 on RDF graph \mathcal{G}_1 , cursor c_2 on RDF graph \mathcal{G}_2 , write access to an empty RDF graph \mathcal{G} , integer n

Out: \mathcal{G} is the cover graph of \mathcal{G}_1 and \mathcal{G}_2

```

1:  $c_1.\text{init}()$   $\triangleright c_1$  at beginning of  $\mathcal{G}_1$ 
2: while  $B_1 = c_1.\text{next}(n)$  do  $\triangleright$  fetch next  $n$   $\mathcal{G}_1$  triples
3:    $c_2.\text{init}()$   $\triangleright c_2$  at beginning of  $\mathcal{G}_2$ 
4:   while  $B_2 = c_2.\text{next}(n)$  do  $\triangleright$  fetch next  $n$   $\mathcal{G}_2$  triples
5:     for all  $T_1 = (s_1, p_1, o_1) \in B_1$  do
6:       for all  $T_2 = (s_2, p_2, o_2) \in B_2$  with  $p_1 = p_2$  do
7:         insert lgau( $T_1, T_2, \text{true}$ ) into  $\mathcal{G}$ 

```

stored in one or several DMSs. It further assumes that the system(s) storing the input RDF graphs \mathcal{G}_1 and \mathcal{G}_2 feature(s) the well-known database mechanism of *cursor* [Garcia-Molina et al., 2009, Ramakrishnan and Gehrke, 2003]. This is for instance the case for RDF graphs stored in relational servers like DB2 [db2, 2012], MySQL [mys, 2017], Oracle [ora, 2017] and PostgreSQL [pos, 2016], or in RDF servers like Jena-TDB [jen, 2017] and Virtuoso [vir, 2017]. Roughly speaking, a cursor is a pointer or iterator on tuples held in a DMS, e.g., stored a relation or computed as the results to a query, that can be used to access these tuples. In particular, a cursor can be used by an application to iteratively traverse all the tuples by fetching n of them at a time.

`lgg4g-dms` uses cursors to proceed similarly to `lgg4g` (remark that lines 5-7 in Algorithm 3 are almost the same as lines 2-4 in Algorithm 2) on pairs of n -triples subsets of \mathcal{G}_1 and of \mathcal{G}_2 , instead of on the whole RDF graphs themselves. It follows that the worst-case number of triples kept in memory by `lgg4g-dms` is $M = (2 \times n) + 1$ at line 7 (i.e., n for B_1 , n for B_2 , and the anti-unifier triple output by `lgau`), with:

$$3 \leq M \leq |\mathcal{G}_1| + |\mathcal{G}_2| + 1.$$

The above lower bound is met for n set to 1, while the upper one is met for n set to $\max(|\mathcal{G}_1|, |\mathcal{G}_2|)$. Importantly, `lgg4g-dms` allows *choosing* the value of n in order to reflect the memory devoted to handling triples. For instance, if one wants to use 4GB of RAM for triples, assuming that any triple fits in less one 1KB (this value is much less when using dictionary encoding [Neumann and Weikum, 2010a], i.e., when triples values are mapped to integers), the value of n can be set to 2M.

This clearly contrasts with the worst-case number of triples kept in memory by `lgg4g`: $M = |\mathcal{G}_1| + |\mathcal{G}_2| + |\mathcal{G}|$ at line 5, with:

$$|\mathcal{G}_1| + |\mathcal{G}_2| \leq M \leq |\mathcal{G}_1| + |\mathcal{G}_2| + (|\mathcal{G}_1| \times |\mathcal{G}_2|).$$

The above lower bound is met when \mathcal{G}_1 and \mathcal{G}_2 have no property in common in their triples (i.e., $|\mathcal{G}| = 0$), while the upper one is met in the unlikely case where \mathcal{G}_1 and \mathcal{G}_2 use a same property in all their triples (i.e., $|\mathcal{G}| = |\mathcal{G}_1| \times |\mathcal{G}_2|$).

4.3.2 Handling huge RDF graphs using MapReduce

Algorithm 4, called `1gg4g-mr`, is a MapReduce (MR) variant of `1gg4g`. MR is a popular massively parallel programming framework [Dean and Ghemawat, 2004], implemented by many large-scale data processing systems, like Hadoop [had, 2016] and Spark [spa, 2017], which orchestrate clusters of compute nodes.

A MR program is organized in successive *jobs*, each of which comprises a *Map task* followed by a *Reduce task*. The Map task consists in reading some input data from the distributed file system¹ of the cluster, so as to partition the data into $\langle \mathbf{k}, \mathbf{v} \rangle$ key-value pairs. Importantly, an MR engine transparently processes the Map task by running *Mapper processes* in parallel on cluster nodes, each process taking care of partitioning a portion of the input data by applying a `Map(key: file, value: data unit)` function on every data unit of a given input file. Key-value pairs thus produced are shuffled across the network, so that *all* pairs with *same* key $\langle \mathbf{k}, \mathbf{v}_1 \rangle \cdots \langle \mathbf{k}, \mathbf{v}_n \rangle$ are shipped to a same compute node. The Reduce task then consists in running *Reducer processes* in parallel, for every distinct key \mathbf{k} received by every compute node. Each process takes care of the set \mathcal{V} of values $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ emitted with key \mathbf{k} , by applying a `Reduce(key: k , values: \mathcal{V})` function, and writing its results in a file. The result of an MR job, comprises the data, stored in a distributed fashion, in all the files output by Reducers.

In `1gg4g-mr`, the `Map` function applies to every $(\mathbf{s}_i, \mathbf{p}_i, \mathbf{o}_i)$ triple of the input RDF graph \mathcal{G}_i stored in file G_i , and produces the corresponding key-value pair $\langle \mathbf{p}_i, (G_i, (\mathbf{s}_i, \mathbf{p}_i, \mathbf{o}_i)) \rangle$, for $i \in [1, 2]$. Hence, all the \mathcal{G}_1 and \mathcal{G}_2 triples with a same key/property p are shipped to the same cluster node. Then, similarly to `1gg4g` at lines 2-4, the `Reduce` functions process, on each node, the set \mathcal{V} of values emitted for every received key p . The least general anti-unification triples obtained at line 4 are stored in the output file $G-p$. At the end of the MR job, the `1gg` \mathcal{G} of \mathcal{G}_1 and \mathcal{G}_2 is stored in the $G-*$ files of the distributed file system, where $*$ denotes any key/property p .

In `1gg4g-mr`, the `Map` function holds at most a single \mathcal{G}_1 or \mathcal{G}_2 triple in memory. In contrast, the worst-case number of triples handled by the `Reduce` function for a given key p is: $M = |\mathcal{G}_1| + |\mathcal{G}_2| + 1$ at line 4. This upper bound is met in the unlikely case where \mathcal{G}_1 and \mathcal{G}_2 use the same property p in all their triples. Similarly to `1gg4g-dms`, this upper bound can set to $M = (2 \times n) + 1$, with $3 \leq M \leq |\mathcal{G}_1| + |\mathcal{G}_2| + 1$, by first splitting the input RDF graphs in k_i files of n \mathcal{G}_i triples (files $G_i^1, \dots, G_i^{k_i}$), and then by processing every pair of n -triples of \mathcal{G}_1 and \mathcal{G}_2 files with an MR job (i.e., with $k_1 \times k_2$ jobs), instead of a single MR job for the entire two input RDF graphs.

Finally, to take into account RDF entailment, the input RDF graphs \mathcal{G}_1 and \mathcal{G}_2 can be saturated before being stored in the MR cluster, by using standard (centralized) techniques, or within the MR cluster, by using MR-based saturation techniques [Urbani et al., 2009, Urbani et al., 2012]. Also, it is worth noting that RDF

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

Algorithm 4 Cover graph of two RDF graphs: `lgg4g-mr`

In: file G_1 for RDF graph \mathcal{G}_1 , file G_2 for RDF graph \mathcal{G}_2

Out: \mathcal{G} is the cover graph of \mathcal{G}_1 and \mathcal{G}_2 , stored in G^* files

Map(key: file G_i , value: triple $T_i = (s_i, p_i, o_i)$)

1: `emit`($\langle p_i, (G_i, T_i) \rangle$)

Reduce(key: p , values: set \mathcal{V} of values emitted for key p)

1: $f \leftarrow \text{open}(G-p)$

2: **for all** $(G_1, T_1 = (s_1, p, o_1)) \in \mathcal{V}$ **do**

3: **for all** $(G_2, T_2 = (s_2, p, o_2)) \in \mathcal{V}$ **do**

4: $f.write(\text{lga}(T_1, T_2, \text{true}))$

5: `close`(f)

graphs, thus `lggs` of RDF graphs, stored in an MR cluster can be queried with MR-based SPARQL engines [Husain et al., 2011, Lee and Liu, 2013, Papailiou et al., 2014, Goasdoué et al., 2015].

4.4 `lggs` of BGPQs

Following Definition 9, Algorithm 5, called `lgg4q`, builds and returns either the cover query q of two input BGPQs q_1 and q_2 if it exists, or the \perp symbol otherwise.

Algorithm 5 Cover query of two BGPQs: `lgg4q`

In: BGPQs q_1 and q_2

Out: Cover query q of q_1 and q_2 if it exists, else \perp

1: $head(q) \leftarrow \text{lga}(head(q_1), head(q_2), \text{false})$

2: $body(q) \leftarrow \emptyset$

3: **for all** $T_1 = (s_1, p_1, o_1) \in \mathcal{G}_1$ **do**

4: **for all** $T_2 = (s_2, p_2, o_2) \in \mathcal{G}_2$ **do**

5: $body(q) \leftarrow body(q) \cup \{\text{lga}(T_1, T_2, \text{false})\}$

6: **if** q is a well-formed BGPQ **then**

7: **return** q

8: **else**

9: **return** \perp

\triangleright error: a cover query does not exist

At line 1, `lgg4q` computes the head of q as the least general anti-unifier of the heads of q_1 and q_2 (first item in Definition 9). Then, at lines 3-5, `lgg4q` computes the body of q , which comprises a least general anti-unifier of every pair of q_1 body and q_2 body triples (second item in Definition 9). Finally, `lgg4q` checks whether q is a *well-formed* BGPQ, i.e., if its answer variables occur in its body triples, otherwise there is no cover query of q_1 and q_2 , and also no `lgg` of q_1 and q_2 (Theorems 9 and 10).

Therefore, given two BGPQs q_1 and q_2 , a call `lgg4q(q1, q2)` produces the cover query-based `lgg` of q_1 and q_2 ignoring RDF entailment and extra ontological constraints,

whenever it exists (Theorem 9), while a call $\mathbf{lgg4q}(q_1^\infty, q_2^\infty)$ produces the cover query-based \mathbf{lgg} of q_1 and q_2 taking into account the set of RDF entailment rules at hand, as well as a set of extra ontological constraints, whenever it exists (Theorem 10). In the latter case, the saturated input BGPQs can be computed using the set characterization of their triples discussed in Section 3.2.

4.5 Conclusion

In this chapter, we provided algorithms in order to compute cover graph-based \mathbf{lggs} of RDF graphs (Definition 3, Algorithm 3) and cover query-based \mathbf{lggs} of BGPQs (Definition 9, Algorithm 5). As input RDF graphs may not fit in memory, we also provided algorithms depending on whether they can fit in a data management system (Algorithm 3) or in a MapReduce cluster (Algorithm 4).

Chapter 5

Experiments

Contents

5.1	Introduction	59
5.2	Gain in precision metrics	60
5.2.1	The case of RDF graphs	60
5.2.2	The case of BGPQs	61
5.3	Experimental setting	62
5.3.1	Software	62
5.3.2	Hardware	62
5.3.3	Datasets	62
5.4	lggs of RDF graphs	65
5.5	lggs of BGPQs	66
5.6	Conclusion	69

5.1 Introduction

In the preceding Chapter, we provided solutions for learning an **lgg** of RDF graphs and of BGPQs, faithful to the RDF and SPARQL W3C recommendations. Notably, we neither restrict the structure nor the semantics of RDF graphs and of BGPQs, while the literature (see Chapter 6) *(i)* imposes structural restrictions and *(ii)* only uses simple entailment (\models) between RDF graphs and between BGPQs, to define **lggs**, hence ignores their saturations, which are essential to consider their full semantics, in particular in presence of ontological knowledge formalized as RDFS constraints. In our experiments, we focus on showing how much more precise **lggs** are when entailment between RDF graphs ($\models_{\mathcal{R}}$) and entailment between BGPQs w.r.t. background knowledge ($\models_{\mathcal{R},\mathcal{O}}$) are utilized instead of just simple entailment (\models).

In Section 5.2, we first establish metrics to quantify the gain in precision that using non-simple entailment yields when learning **lggs** of RDF graphs and of BGPQs. Then, in Section 5.3, we describe our experimental setting. Finally, we analyze within this

setting the gain in precision of **lggs** of RDF graphs in Section 5.4, and of BGPQs in Section 5.5, depending on the entailment relation under consideration.

5.2 Gain in precision metrics

5.2.1 The case of RDF graphs

The next result shows that measuring the gain in precision for an **lgg** of RDF graphs when standard entailment is used instead of simple entailment, amounts to *measuring an entailment distance* through the standard relation of entailment between RDF graphs ($\models_{\mathcal{R}}$).

Proposition 5. *Let $\mathcal{G}_1, \dots, \mathcal{G}_n$ be n RDF graphs, \mathcal{G}_{lgg} an **lgg** of $\mathcal{G}_1, \dots, \mathcal{G}_n$ w.r.t. the empty set of entailment rules (i.e., defined using simple entailment \models between RDF graphs) and $\mathcal{G}_{\text{lgg}}^{\mathcal{R}}$ an **lgg** of $\mathcal{G}_1, \dots, \mathcal{G}_n$ w.r.t. any set \mathcal{R} of entailment rules (i.e., defined using standard entailment $\models_{\mathcal{R}}$ between RDF graphs). Then, $\mathcal{G}_{\text{lgg}}^{\mathcal{R}} \models_{\mathcal{R}} \mathcal{G}_{\text{lgg}}$ holds.*

Proof. Since all the **lggs** of some given RDF graphs are equivalent (Theorem 1): (i) \mathcal{G}_{lgg} is *equivalent* to the cover graph-based **lgg** \mathcal{G} of $\mathcal{G}_1, \dots, \mathcal{G}_n$, i.e., $\mathcal{G}_{\text{lgg}} \equiv_{\mathcal{R}} \mathcal{G}$ holds, and (ii) $\mathcal{G}_{\text{lgg}}^{\mathcal{R}}$ is *equivalent* to the cover graph-based **lgg** \mathcal{G}' of the saturations with \mathcal{R} of $\mathcal{G}_1, \dots, \mathcal{G}_n$, i.e., $\mathcal{G}_{\text{lgg}}^{\mathcal{R}} \equiv_{\mathcal{R}} \mathcal{G}'$ holds. These cover graph-based **lggs** can be computed based on Proposition 1 when $n > 2$. Further, *by definition* of a cover graph (Definition 3), \mathcal{G} is a subset of \mathcal{G}' , thus (iii) $\mathcal{G}' \models \mathcal{G}$ holds. Therefore, from (i), (ii) and (iii), $\mathcal{G}_{\text{lgg}}^{\mathcal{R}} \models_{\mathcal{R}} \mathcal{G}_{\text{lgg}}$ holds. \square

From the above result, and by the definition of an **lgg**, it immediately follows that for each input RDF graph $\mathcal{G}_{1 \leq i \leq n}$, $\mathcal{G}_i \models_{\mathcal{R}} \mathcal{G}_{\text{lgg}}^{\mathcal{R}} \models_{\mathcal{R}} \mathcal{G}_{\text{lgg}}$ holds. Further, whenever the **lggs** \mathcal{G}_{lgg} and $\mathcal{G}_{\text{lgg}}^{\mathcal{R}}$ are the *cover graph-based ones* computed from $\mathcal{G}_1, \dots, \mathcal{G}_n$ and from their saturations respectively, then: if $\mathcal{G}_i \models_{\mathcal{R}}^{\phi_i} \mathcal{G}_{\text{lgg}}^{\mathcal{R}}$ holds then $\mathcal{G}_i \models_{\mathcal{R}}^{\phi_i} \mathcal{G}_{\text{lgg}}$ also holds, i.e., if $\mathcal{G}_i^{\infty} \models^{\phi_i} \mathcal{G}_{\text{lgg}}^{\mathcal{R}}$ holds then $\mathcal{G}_i^{\infty} \models^{\phi_i} \mathcal{G}_{\text{lgg}}$ also holds, since by definition and construction of these cover graph-based **lggs**: $\mathcal{G}_{\text{lgg}} \subseteq \mathcal{G}_{\text{lgg}}^{\mathcal{R}}$ holds. Interestingly, in this case, the RDF graphs $[\mathcal{G}_{\text{lgg}}]_{\phi_i}$ and $[\mathcal{G}_{\text{lgg}}^{\mathcal{R}}]_{\phi_i}$ are such that $[\mathcal{G}_{\text{lgg}}]_{\phi_i} \subseteq [\mathcal{G}_{\text{lgg}}^{\mathcal{R}}]_{\phi_i} \subseteq \mathcal{G}_i^{\infty}$ holds and they represent the triples from \mathcal{G}_i and from \mathcal{G}_i^{∞} respectively, that participated in anti-unifications to beget the cover graph-based **lggs** \mathcal{G}_{lgg} and $\mathcal{G}_{\text{lgg}}^{\mathcal{R}}$ respectively. In addition, remark that for the homomorphism ϕ_i that directly follows from our naming of $\mathcal{G}_{\text{lgg}}^{\mathcal{R}}$ blank nodes¹, $[\mathcal{G}_{\text{lgg}}]_{\phi_i}$ and $[\mathcal{G}_{\text{lgg}}^{\mathcal{R}}]_{\phi_i}$ are clearly *maximal* as they result, by construction, of all possible anti-unifications of triples. Therefore, when ϕ_i is maximal, the saturations of these RDF graphs $([\mathcal{G}_{\text{lgg}}]_{\phi_i})^{\infty}$ and $([\mathcal{G}_{\text{lgg}}^{\mathcal{R}}]_{\phi_i})^{\infty}$ are such that $([\mathcal{G}_{\text{lgg}}]_{\phi_i})^{\infty} \subseteq ([\mathcal{G}_{\text{lgg}}^{\mathcal{R}}]_{\phi_i})^{\infty} \subseteq \mathcal{G}_i^{\infty}$ holds and they represent the subsets of *all* explicit and implicit \mathcal{G}_i triples that are generalized by those of \mathcal{G}_{lgg} and $\mathcal{G}_{\text{lgg}}^{\mathcal{R}}$ respectively.

1. From the definition of a cover graph \mathcal{G} of the RDF graphs \mathcal{G}_1 and \mathcal{G}_2 (Definition 1), the homomorphisms ϕ_1 and ϕ_2 defined as if $b_{t_1 t_3}$ (resp. $b_{t_2 t_4}$) is a \mathcal{G} blank node then the homomorphisms $\phi_1(b_{t_1 t_3}) = t_1$ and $\phi_2(b_{t_1 t_3}) = t_3$ (resp. $\phi_1(b_{t_2 t_4}) = t_2$ and $\phi_2(b_{t_2 t_4}) = t_4$), are clearly such that $\mathcal{G}_1 \models^{\phi_1} \mathcal{G}$ and $\mathcal{G}_2 \models^{\phi_2} \mathcal{G}$ hold.

From the above observation, we measure the *gain in precision of the lgg* $\mathcal{G}_{1\text{gg}}^{\mathcal{R}}$ over the lgg $\mathcal{G}_{1\text{gg}}$ w.r.t. the input RDF graph $\mathcal{G}_{1 \leq i \leq n}$, noted ρ_i , as the percentage of explicit and implicit \mathcal{G}_i triples that $\mathcal{G}_{1\text{gg}}^{\mathcal{R}}$ generalizes while $\mathcal{G}_{1\text{gg}}$ does not:

$$\rho_i = \frac{|([\mathcal{G}_{1\text{gg}}^{\mathcal{R}}]_{\phi_i})^\infty| - |([\mathcal{G}_{1\text{gg}}]_{\phi_i})^\infty|}{|\mathcal{G}_i^\infty|}.$$

Finally, we measure the *gain in precision of the lgg* $\mathcal{G}_{1\text{gg}}^{\mathcal{R}}$ over the lgg $\mathcal{G}_{1\text{gg}}$ w.r.t. the input RDF graphs $\mathcal{G}_1, \dots, \mathcal{G}_n$, noted ρ , as the average of the above ρ_1, \dots, ρ_n :

$$\rho = \frac{\sum_{i=1}^n \rho_i}{n}.$$

5.2.2 The case of BGPQs

Similarly to the case of RDF graphs, we establish:

Proposition 6. *Let \mathcal{R} be a set of entailment rules, \mathcal{O} a set of RDFS statements, and q_1, \dots, q_n BGPQs with same arity. An lgg $q_{1\text{gg}}$ of q_1, \dots, q_n (i.e., defined using simple entailment \models) and an lgg $q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}}$ of q_1, \dots, q_n w.r.t. \mathcal{R} and \mathcal{O} (i.e., defined using our novel entailment relation $\models_{\mathcal{R}, \mathcal{O}}$) are such that: $q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}} \models_{\mathcal{R}, \mathcal{O}} q_{1\text{gg}}$ holds.*

Proof. Since all the lgg of some given BGPQs are equivalent (Theorem 8): (i) $q_{1\text{gg}}$ is equivalent to the cover query-based lgg q of q_1, \dots, q_n , i.e., $q_{1\text{gg}} \equiv_{\mathcal{R}} q$, and (ii) $q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}}$ is equivalent to the cover query-based lgg q' of the saturations with \mathcal{R} of q_1, \dots, q_n w.r.t. \mathcal{O} , i.e., $q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}} \equiv_{\mathcal{R}} q'$. These cover query-based lgg can be computed based on Proposition 3 when $n > 2$. Further, by definition of a cover query (Definition 9), q and q' have the same heads and the body of q is a subset of that q' , thus (iii) $q' \models q$ holds. Therefore, from (i), (ii) and (iii), $q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}} \models_{\mathcal{R}, \mathcal{O}} q_{1\text{gg}}$ holds. \square

From the above result, we devise a new gain in precision metric for the lgg of BGPQs, because the adaptation of the above one for the lgg of RDF graphs does not make sense here. Indeed, the above metric measures on average how much more \mathcal{G}_i triples participate in anti-unifications when standard entailment ($\models_{\mathcal{R}}$) is used instead of simple entailment (\models) when computing a cover graph-based lgg of RDF graphs. However, in the case of BGPQs, all q_i and all $q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}}$ triples participate in anti-unifications, because an unknown value is allowed in the property position of a BGPQ triples while this is not the case for RDF graph triples (recall that the property must be a URI). As a result, adapting the above metric to lgg of BGPQs would only reflect the size increase between $q_{1\text{gg}}$ and $q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}}$ that the saturation of $q_{1\text{gg}}$ w.r.t. \mathcal{R} and \mathcal{O} produces.

Instead, we rely on the fact that from the *query answering* point of view, using Property 4 (Section 1.3) and the above Proposition 6, $q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}}(\mathcal{G}) \subseteq q_{1\text{gg}}(\mathcal{G})$ holds for any RDF graph \mathcal{G} , and clearly the more $q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}}$ is specific w.r.t. $q_{1\text{gg}}$ through $\models_{\mathcal{R}}$, the smaller the subset $q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}}(\mathcal{G})$ of $q_{1\text{gg}}(\mathcal{G})$ is, i.e., the smaller $|q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}}(\mathcal{G})|$ is w.r.t. $|q_{1\text{gg}}(\mathcal{G})|$. Therefore, to measure the semantic distance between $q_{1\text{gg}}^{\mathcal{R}, \mathcal{O}}$ and $q_{1\text{gg}}$ through $\models_{\mathcal{R}}$, we define the *gain*

in precision ρ that standard entailment endowed with background knowledge ($\models_{\mathcal{R},\mathcal{O}}$) yields over simple entailment (\models) w.r.t. query answering as the percentage of $q_{1\text{gg}}$ answers against \mathcal{G} that are not $q_{1\text{gg}}^{\mathcal{R},\mathcal{O}}$ against \mathcal{G} :

$$\rho = 1 - \frac{|q_{1\text{gg}}^{\mathcal{R},\mathcal{O}}(\mathcal{G})|}{|q_{1\text{gg}}(\mathcal{G})|}.$$

5.3 Experimental setting

We describe now the setup used to evaluate our technical contributions.

5.3.1 Software

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.

We used **Jena** to compute the *saturation of an RDF graph*, against which queries must be evaluated to obtain their *complete* answer sets (Chapter 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 5): we rely on Jena’s saturation, union and difference operators to compute $q_{\infty}^{\mathcal{O}}$ ’s body.

We used **PostgreSQL** to evaluate *SQLized BGPQs* against a *saturated* RDF graph stored in a `TripleDBpedia(s,p,o)` table. The table is indexed by all permutations of the `s`, `p`, `o` columns, leading to a total of 6 indexes. This indexing choice is inspired by [Neumann and Weikum, 2010b, Weiss et al., 2008], to give PostgreSQL efficient query evaluation opportunities. There exists 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` [Bursztyn et al., 2016], or the elaborate layout of [Bornea et al., 2013] 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.

5.3.2 Hardware

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

Next, all measured times on this hardware are averaged over 5 warm runs and are in milliseconds.

5.3.3 Datasets

We selected two datasets to conduct our experiments, the *real DBpedia dataset* [Lehmann et al., 2015] and a *synthetic dataset on LUBM universities* [Guo et al., 2005].

The RDF graph $\mathcal{G}_{\text{LUBM}}$, which we generated with the LUBM data generator², com-

2. <http://swat.cse.lehigh.edu/projects/lubm/>

prises 884k triples before saturation, including a subset $\mathcal{O}_{\text{LUBM}}$ of 242 RDFS constraints. The saturation of $\mathcal{G}_{\text{LUBM}}$ comprises 1.08M triples. The total time for the generation of $\mathcal{G}_{\text{LUBM}}$, its saturation and its loading into the PostgreSQL `TripleLUBM(s,p,o)` table is about 1 hour.

To build the RDF graph $\mathcal{G}_{\text{DBpedia}}$, we picked four complementary DBpedia files³. The RDF graph $\mathcal{G}_{\text{DBpedia}}$ comprises 41.18M triples, including a subset $\mathcal{O}_{\text{DBpedia}}$ of 30.31k RDFS constraints. The saturation of $\mathcal{G}_{\text{DBpedia}}$ comprises 78.14M triples. The total time of the saturation of $\mathcal{G}_{\text{DBpedia}}$ and its loading into the PostgreSQL `Triple(s,p,o)` table is about 24 hours.

In our experiments, we always used the subset of standard RDF entailment rules shown in Table 1.2 (Chapter 1), which fully allows exploiting RDFS ontological constraints.

Test RDF Graphs Our experiments with RDF graphs rely on 9 subgraphs of $\mathcal{G}_{\text{DBpedia}}$ which contain 397 triples from $\mathcal{O}_{\text{DBpedia}}$, related to the selected subgraphs. Their characteristics are shown in Table 5.1 (top) and are further detailed in Appendix A, page 85. The 397 triples from $\mathcal{O}_{\text{DBpedia}}$ are listed in Appendix D, page 93. We distinguish two subsets of RDF graphs: G_1 - G_4 , Table 5.1 (left), are heterogeneous in the sense that they differ both on their structure and on the kind of information they represent, hence use distinct classes, properties and URI values, while G_5 - G_9 , Table 5.1 (right), are more homogeneous and only differ in some properties and URI values.

Table 5.1 shows some properties of our test RDF graphs. Line 2 of Table 5.1 gives the number of triples which are not ontological constraints as all contain 397 ontological constraints. Line 3 of Table 5.1 provides the number of triples in the saturated RDF graphs that are not ontological constraints (whose size is precisely 1067 triples). The size of our RDF graph tests varies from $4 + 397$ to $6 + 397$ (line 3, Table 5.1) while their saturations (line 4, Table 5.1) vary from $16 + 1067$ to $25 + 1067$ triples. The size increase due to saturation is $\times 2.7$ on average; the size increase of the non ontological subsets varies from $\times 3.4$ (G_2) to $\times 4.8$ (G_9). The saturation of a test RDF graphs requires 75.44 ms on average.

DBpedia graph $G_{1 \leq i \leq 9}$:	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9
$ G_i $'s shape	Graph	Graph	Tree	Graph	Graph	Graph	Graph	Graph	Graph
$ G_i \setminus \mathcal{O}_{\text{DBpedia}} $	4	5	4	6	6	4	6	5	5
$ G_i^\infty \setminus \mathcal{O}_{\text{DBpedia}}^\infty $	16	17	19	22	23	19	25	21	24
Time to compute G_i^∞	66	88	90	92	66	68	63	54	92

Table 5.1 – Characteristics of our test graphs (top) and of their saturations (bottom); times are in ms.

Test Queries We used queries for the two datasets previously presented, which were executed on the entire saturations of $\mathcal{G}_{\text{LUBM}}$ and $\mathcal{G}_{\text{DBpedia}}$ respectively.

3. 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.

LUBM query $Q_{1 \leq i \leq 9}$:	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
Q_i 's shape	star	star	graph	graph	star	tree	graph	star	star
$ body(Q_i) $	3	3	4	3	2	4	6	2	3
number of URI/variable occurrence in Q	5/4	5/4	5/7	4/5	3/3	6/6	7/11	2/4	5/4
$ Q_i(\mathcal{G}_{LUBM}) $	123	41	869	0	269	41 751	79	14 252	16
$ body(Q_i \mathcal{O}_{LUBM}^\infty) $	8	9	11	7	6	8	14	10	8
Time to compute $Q_i \mathcal{O}_{LUBM}^\infty$	24	23	23	21	22	22	21	27	22

Table 5.2 – Characteristics of our test BGPQs (top) and of their saturations w.r.t. LUBM constraints (bottom); times are in ms.

DBpedia 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}_{DBpedia}) $	77	0	41 695	13	6	0	1	0
$ body(Q_i \mathcal{O}_{DBpedia}^\infty) $	16	19	19	23	16	23	23	23
Time to compute $Q_i \mathcal{O}_{DBpedia}^\infty$	666	643	677	734	681	706	697	736

Table 5.3 – Characteristics of our test BGPQs (top) and of their saturations w.r.t. DBpedia constraints (bottom); times are in ms.

LUBM queries: we borrowed from [Bursztyrn et al., 2015] a set of 9 BGPQs which are described in Appendix B, page 89. Their characteristics are displayed in Table 5.2 (top): queries have a variety of structural aspects (lines 1 and 3) and body size (line 2) and differs in the answer numbers (line 4).

DBpedia queries: we defined 42 test BGPQs among which we picked 8 representative ones. These 8 queries are described in Appendix C, page 91. Their characteristics are displayed in Table 5.3 (top). Similarly to our selected LUBM queries, DBpedia queries have a variety of structural aspects (lines 1 and 3) and body size (line 2) and differs in the answer numbers (line 4).

We consider two subsets of DBpedia queries: Q_1 - Q_4 , Table 5.3 (left), are heterogeneous, while Q_4 - Q_8 , in Table 5.3 (right), are homogeneous.

The Saturations of our test queries w.r.t. **background knowledge**, \mathcal{O}_{LUBM} and $\mathcal{O}_{DBpedia}$ respectively, are presented Tables 5.2 (bottom) and 5.3 (bottom). They show the size of the saturated test queries and the time to compute them. Enriching our test queries using the LUBM and DBpedia ontological constraints augments their size: from $\times 2$ for Q_6 up to $\times 5$ for Q_8 in the LUBM case; from $\times 3.16$ for Q_2 up to $\times 4.75$ for Q_3 in the DBpedia case. The query saturation time is rather fast with LUBM, 23 ms on average, while it is 692 ms on average with DBpedia. Saturation times have been multiplied by 30 from LUBM to DBpedia, since the latter has $\times 125$ more constraints than the former.

5.4 1ggs of RDF graphs

We report now on the experiments we made with our test RDF graphs.

Table 5.4 (lines 1 and 2) shows that cover graph-based 1ggs of test RDF graphs are computed quickly whether or not they are saturated: 4 to 6 ms when they are not saturated, and 8 to 15 ms when they are saturated.

Line 3, Table 5.4, shows the gain brought by considering standard entailment w.r.t. simple entailment. To fully assess this gain, as the number of own triples of each test graph is low compared to those of the size of their ontological part, we slightly modify our metric defined in section 5.2.1 so that the ontological constraints are not considered: as all input test RDF graphs, hence their 1gg share these constraints. Thus, line 3 Table 5.4 points out that using standard entailment when computing an 1gg offers a significant precision gain for heterogenous graphs (from 42.46% to 61.12%), and a more limited but notable precision gain for homogeneous graphs (from 0% to 19.89%).

The gain of 0% for graphs G_6 and G_8 is explained by the fact their structures are the same. They differ only on subject and object URIs and the 4 distinct triples of G_6 have the same property URIs as 4 triples among 5 from G_8 .

For heterogenous test RDF graphs, the gain is considerable for G_3 and G_4 (61.12%): they have only one common property; nevertheless almost all the remaining properties of G_3 have a common super-property with one of G_4 , which are made explicit through saturation.

1gg of 2 DBpedia graphs:	G_1G_2	G_1G_3	G_1G_4	G_2G_4	G_3G_4	G_5G_6	G_5G_9	G_6G_7	G_6G_8	G_7G_8	G_8G_9
Time to compute G_{1gg}	5	5	4	5	5	5	6	6	5	5	5
Time to compute G_{1gg}^R	8	9	11	15	13	8	14	8	8	9	10
Gain in precision	42.46	47.70	49.15	47.86	61.12	2.17	2.17	19.89	0	19.14	13.39

Table 5.4 – Characteristics of cover graph-based 1ggs of 2 test graphs, w/ or w/o saturation w.r.t. DBpedia RDFS constraints; times are in ms.

1gg of 3 DBpedia graphs:	$G_1G_2G_3$	$G_1G_2G_4$	$G_1G_3G_4$	$G_2G_3G_4$	$G_5G_6G_7$	$G_5G_6G_8$	$G_5G_6G_9$	$G_6G_7G_8$	$G_7G_8G_9$
Time to compute G_{1gg}	10	10	11	12	11	12	11	11	11
Time to compute G_{1gg}^R	30	41	39	41	33	42	37	49	52
Gain in precision	47.48	50.41	48.46	51.65	19.06	1.44	19.39	29.13	16.92

Table 5.5 – Characteristics of cover graph-based 1ggs of 3 test graphs, w/ or w/o saturation w.r.t. DBpedia RDFS constraints; times are in ms.

We have also computed 1ggs of k test RDF graphs, with $2 < k \leq 4$. Table 5.5 and Table 5.6 show the time to compute them, and the gain in precision of using standard entailment instead of simple entailment.

As defined in Proposition 1, Chapter 2, 1gg of k test RDF graphs are computed as a sequence of $k - 1$ cover graph-based 1ggs of two RDF graphs. For example, the 1gg $G_1G_2G_4$ (Table 5.5) is the cover graph-based 1gg of the cover graph-based 1gg G_1G_2 (Table 5.4) and the test RDF graphs graph G_4 .

lgg of 4 DBpedia graphs:	$G_1G_2G_3G_4$	$G_5G_6G_7G_9$	$G_5G_6G_8G_9$	$G_6G_7G_8G_9$
Time to compute $G_{1\text{lgg}}$	15	17	17	18
Time to compute $G_{1\text{lgg}}^R$	142	151	240	194
Gain in precision	48.11	21.54	19.31	20.81

Table 5.6 – Characteristics of cover graph-based lgggs of 4 test graphs, w/ or w/o saturation w.r.t. DBpedia RDFS constraints; times are in ms.

Table 5.5 and Table 5.6 (lines 1 and 2) outline that the computation of a cover graph-based lgggs of k test RDF graphs is rather fast when simple entailment is used (10 to 18 ms), and a little slower when standard entailment is applied (30 to 240 ms). These latter measures represent the time required to compute the entire sequence of $k - 1$ cover graph-based lgggs. Table 5.5 and Table 5.6 (line 3) highlight that using standard entailment also significantly increases the precision of some lgggs, from a small gain of 1.44% for $G_5G_6G_8$ up to an important gain of 51.65% for $G_2G_3G_4$. Therefore, even for lgg of more than 2 graphs, using standard entailment instead of simple entailment enables to get more precise lgg.

5.5 lgggs of BGPQs

We report now on the experiments we made with our test BGPQs.

LUBM. Table 5.7 (lines 1 and 3) shows that cover query-based lgggs of test queries are always computed fast whether or not we consider the LUBM ontological constraints: 1 to 4 ms when they are ignored, and 5 to 10 ms when they are considered. In the latter case, overall, it takes between 50 and 59 ms to compute an lgg in the worst case (i.e., when the two saturated test queries are computed *in sequence* before computing their cover query). Table 5.7 (lines 2 and 4) shows that ignoring background knowledge significantly increases the number of answers for some lgggs, from a small $\times 1.32$ for Q_2Q_8 up to a striking $\times 73.39$ for Q_2Q_4 , with a significant average of $\times 16.45$. This translates into the precision gains shown at line 5 (Table 5.7): 74.80% overall, 98.63% for queries Q_2 and Q_4 , and 24.49% for Q_2 and Q_8 . lgggs with same number of answers in Table 5.7 were found either equivalent or almost equivalent so that their semantic difference is not visible on the LUBM dataset, e.g., the lgggs of Q_2Q_4 , Q_1Q_8 , Q_5Q_8 and Q_8Q_9 with 1 048 360 answers (i.e., ignoring ontological constraints) are asking for all the (distinct) subject/object pairs in triples in the LUBM RDF graph; the lgggs of Q_8Q_9 , Q_2Q_8 with 56 358 answers (i.e., considering ontological constraints) ask respectively for faculties with some degree from some university and for employees with some doctoral degree from some university: they have the same number of answers in our LUBM RDF graph because all faculties have some doctoral degree and all employees with some doctoral degree are faculties.

We have also computed lgggs of k test queries where $2 < k \leq 4$. Table 5.8 and Table 5.9 show the obtained cover query-based lgggs. As defined in Proposition 3, chapter 3, lgg of k test queries are computed as a sequence of $k - 1$ cover query-based lgggs of two queries. Table 5.8 and Table 5.9 (lines 1 and 3) point out that the

lgg of 2 LUBM queries:	Q_1Q_4	Q_2Q_4	Q_1Q_8	Q_2Q_8	Q_5Q_8	Q_1Q_9	Q_8Q_9	Q_3Q_6	Q_3Q_7
Time to compute q_{lgg}	1	2	1	1	1	3	2	2	4
$ q_{lgg}(\mathcal{G}_{LUBM}) $	30 963	1 048 360	1 048 360	74 643	1 048 360	253 443	1 048 360	6 937 472	57 774
Time to compute $q_{lgg}^{O_{LUBM}}$	5	6	6	7	10	6	6	6	8
$ q_{lgg}^{O_{LUBM}}(\mathcal{G}_{LUBM}) $	14 285	14 285	56 358	56 358	185 124	19 053	56 358	520 365	34 852
Gain in precision	53.86	98.63	94.62	24.49	82.34	92.48	94.62	92.49	39.67

Table 5.7 – Characteristics of cover query-based lggs of test queries, w or w/o using the LUBM RDFS constraints; times are in ms.

lgg of 3 LUBM queries:	$Q_1Q_2Q_4$	$Q_1Q_8Q_9$	$Q_2Q_8Q_9$	$Q_1Q_4Q_8$	$Q_1Q_4Q_9$	$Q_2Q_4Q_8$	$Q_1Q_5Q_8$	$Q_5Q_8Q_9$
Time to compute q_{lgg}	6	6	5	4	7	5	4	3
$ q_{lgg}(\mathcal{G}_{LUBM}) $	1 048 060	1 048 060	1 048 060	1 048 060	253 443	1 048 060	1 048 060	1 048 060
Time to compute $q_{lgg}^{O_{LUBM}}$	11	13	11	11	12	13	15	14
$ q_{lgg}^{O_{LUBM}}(\mathcal{G}_{LUBM}) $	14 285	120 225	120 225	33 305	19 053	33 305	120 225	120 225
Gain in precision	98.63	88.52	88.52	96.82	92.48	96.82	88.52	88.52

Table 5.8 – Characteristics of cover query-based lggs of 3 test queries, w or w/o using the LUBM RDFS constraints; times are in ms.

computation of a cover query-based lggs of k test queries is rather fast whether or not we consider the LUBM ontological constraints: 3 to 10 ms when they are ignored, and 11 to 23 ms when they are considered. These latter times represent the time required to compute the entire sequence of $k - 1$ cover query-based lggs. Table 5.8 and Table 5.9 (lines 2 and 4) highlight that ignoring background knowledge also significantly increases the number of answers for some lggs, from a small $\times 8.71$ for $Q_1Q_2Q_8Q_9$ up to a striking $\times 73.36$ for $Q_1Q_2Q_4$. This is reflected by the precision gains shown at line 5 (Table 5.8 and Table 5.9): 92.35% overall for lgg of 3 test queries, 91.09% overall for lgg of 4 test queries. Thus, even for lgg of more than 2 queries, taking into account LUBM ontological constraints enables to get more precise lgg. We remark that lggs of k queries ignoring background knowledge yield to an equivalent cover query-based lgg as soon as k is 3 (see Table 5.8, line 2, lgg of 3 test queries having 1 048 060 answers), while lggs of k queries taking into account LUBM ontological constraints begin to become almost all equivalent when k reaches 4 (see Table 5.9, line 4, lgg of 4 test queries having 120 225 answers).

lgg of 4 LUBM queries:	$Q_1Q_2Q_4Q_8$	$Q_1Q_2Q_4Q_9$	$Q_1Q_2Q_8Q_9$	$Q_1Q_4Q_5Q_8$	$Q_1Q_5Q_8Q_9$	$Q_2Q_4Q_8Q_9$	$Q_2Q_5Q_8Q_9$
Time to compute q_{lgg}	7	10	8	5	7	6	6
$ q_{lgg}(\mathcal{G}_{LUBM}) $	1 048 060	1 048 060	1 048 060	1 048 060	1 048 060	1 048 060	1 048 060
Time to compute $q_{lgg}^{O_{LUBM}}$	17	19	18	23	18	18	20
$ q_{lgg}^{O_{LUBM}}(\mathcal{G}_{LUBM}) $	33 305	19 053	120 225	120 225	120 225	120 225	120 225
Gain in precision	96.82	98.18	88.52	88.52	88.52	88.52	88.52

Table 5.9 – Characteristics of cover query-based lggs of 4 test queries, w or w/o using the LUBM RDFS constraints; times are in ms.

l _{gg} of 2 DBpedia queries:	Q_1Q_2	Q_1Q_3	Q_1Q_4	Q_2Q_3	Q_4Q_7	Q_4Q_8	Q_5Q_6	Q_5Q_7	Q_7Q_8
Time to compute $q_{l_{gg}}$	3	3	5	4	5	6	5	6	5
$ q_{l_{gg}}(\mathcal{G}_{DBpedia}) $	477 455	34 747 102	34 901 117	60 356 807	34	27	1221	35	70
Time to compute $q_{l_{gg}}^{\mathcal{O}_{DBpedia}}$	13	14	14	15	15	14	14	17	18
$ q_{l_{gg}}^{\mathcal{O}_{DBpedia}}(\mathcal{G}_{DBpedia}) $	10 637	7 874 768	456 690	7 874 768	34	13	780	34	36
Gain in precision	97.77	77.33	98.69	86.95	0	51.85	36.11	2.85	48.57

Table 5.10 – Characteristics of cover query-based l_{ggs} of test queries, w/ or w/o using the DBpedia RDFS constraints; times are in ms.

l _{gg} of 3 DBpedia queries:	$Q_1Q_2Q_3$	$Q_1Q_2Q_4$	$Q_1Q_3Q_4$	$Q_2Q_3Q_4$	$Q_4Q_7Q_8$	$Q_5Q_7Q_8$
Time to compute $q_{l_{gg}}$	5	6	6	6	10	9
$ q_{l_{gg}}(\mathcal{G}_{DBpedia}) $	34 747 102	34 901 117	34 901 117	34 901 117	70	1 977
Time to compute $q_{l_{gg}}^{\mathcal{O}_{DBpedia}}$	22	23	23	25	27	29
$ q_{l_{gg}}^{\mathcal{O}_{DBpedia}}(\mathcal{G}_{DBpedia}) $	7 874 768	615 339	7 874 779	4 537 824	36	1 701
Gain in precision	77.33	98.23	77.43	86.99	48.57	13.96

Table 5.11 – Characteristics of cover query-based l_{ggs} of 3 test queries, w/ or w/o using the DBpedia RDFS constraints; times are in ms.

l _{gg} of 4 DBpedia queries:	$Q_1Q_2Q_3Q_4$	$Q_4Q_5Q_7Q_8$
Time to compute $q_{l_{gg}}$	10	14
$ q_{l_{gg}}(\mathcal{G}_{DBpedia}) $	34 901 117	1 977
Time to compute $q_{l_{gg}}^{\mathcal{O}_{DBpedia}}$	30	36
$ q_{l_{gg}}^{\mathcal{O}_{DBpedia}}(\mathcal{G}_{DBpedia}) $	7 874 779	1 701
Gain in precision	77.43	13.96

Table 5.12 – Characteristics of cover query-based l_{ggs} of 4 test queries, w/ or w/o using the DBpedia RDFS constraints; times are in ms.

DBpedia. First, as Table 5.10 (lines 1 and 3) shows, the cover query-based l_{ggs} of test queries are always computed fast whether or not the DBpedia ontological constraints are considered: from 3 to 6 ms when ignored, to 13 to 18 ms when considered.

Table 5.10 (lines 2 and 4) also shows that the answer set of an l_{gg} is significantly larger when DBpedia ontological constraints are not taken into account: the size difference goes from a small $\times 1$ for the homogeneous queries Q_4, Q_7 up to a striking $\times 76.42$ for the heterogeneous queries Q_1, Q_4 , with a significant average of $\times 17.50$ ($\times 33.34$ for the heterogeneous queries and $\times 1.65$ for the homogeneous ones). This translates into the precision gains shown at line 5: 55.57% overall, 90.18% for the heterogeneous queries, and 27.88% for the homogeneous ones.

Then, we have also computed l_{ggs} of k DBpedia test queries where $2 < k \leq 4$. Table 5.11 and Table 5.12 display the obtained cover query-based l_{ggs}. Table 5.11 and Table 5.12 (lines 1 and 3) point out that the computation of a cover query-based l_{ggs} of k test graphs is fast whether simple entailment or standard entailment endowed with background knowledge is using (from 5 to 14 ms and from 22 to 36 ms respectively).

These measures denote the time required to compute the entire sequence of $k - 1$ cover query-based **lggs**. Table 5.11 and Table 5.12 (line 5) outline that using entailment w.r.t. background knowledge also significantly increases the precision of some **lggs**, from a small gain of 13.96% for $Q_5Q_7Q_8$ up to a considerable gain of 98.23 for $Q_1Q_2Q_4$. Therefore, even for DBpedia test queries, **lgg** of more than 2 queries which relies on entailment between BGPQs w.r.t. background knowledge lead to more precise **lgg**.

5.6 Conclusion

Our results confirm our claim that *using general entailment between RDF graphs* ($\models_{\mathcal{R}}$) and *entailment between BGPQs w.r.t. background knowledge* ($\models_{\mathcal{R},\mathcal{O}}$) instead of simple entailment (\models) yield more precise **lggs**. Indeed, exploiting ontological constraints help finding *common super-* classes and properties to be used in **lggs** in place of the different ones used in input graphs and queries. When simple entailment is used during graph **lgg** computation, some generalizations are missing while in the case of query **lgg** computation, properties and classes are generalized using *variables*. Therefore, the more heterogeneous input graphs and queries are, the more such common super-classes and properties may be used in their **lgg** computation and the more the gain in precision of their **lgg** is high. For homogeneous input graphs and queries, while less striking, the gain in precision is significant in general, as our experiments show.

Chapter 6

Related Work

Contents

6.1	Introduction	71
6.2	Inductive Logic Programming	71
6.3	Knowledge Representation	72
6.3.1	Description logics	72
6.3.2	Conceptual graphs	74
6.4	Semantic Web	74
6.4.1	RDF	74
6.4.2	SPARQL	75
6.5	Conclusion	76

6.1 Introduction

The problem of computing some least general generalization of a set of descriptions has been introduced by G. Plotkin in the early 70's [Plotkin, 1970, Plotkin, 1971]. Since then, this problem has been investigated within the Inductive Logic Programming (ILP), the Knowledge Representation (KR) and the Semantic Web (SW) fields.

In this chapter, we survey the related works from these research fields, and relate them to the contributions of this thesis.

6.2 Inductive Logic Programming

G. Plotkin introduced the problem of computing an `lgg` in a First Order Logic setting. He first considered clauses as descriptions and θ -subsumption, a restricted form of logical implication now typical of the ILP field, as generalization/specialization relation [Plotkin, 1970]. Then, he also considered finding an `lgg` of clauses in presence of background knowledge formalized as a set of ground literals (positive or negative facts) [Plotkin, 1971]. These two foundational works and extensions to standard implication [Buntine, 1988,

Nienhuys-Cheng and de Wolf, 1996, Nienhuys-Cheng and de Wolf, 1997], constitute the basis of many ILP systems [Raedt, 2008]; recent ILP works build on them [Lisi, 2007, Raedt, 2008, Kuzelka et al., 2012].

The main idea of [Plotkin, 1970, Plotkin, 1971, Buntine, 1988, Nienhuys-Cheng and de Wolf, 1996, Nienhuys-Cheng and de Wolf, 1997] is to compute an **lgg** of two clauses as the clause made of all the possible anti-unifications between the literals of the two clauses; across these anti-unifications, a same pair of literal terms is anti-unified to a same value by assuming given a injective naming function. Further, to take into account background knowledge formalized as ground literals, their **lgg** can to be computed as described above, not from the input clauses themselves, but from their saturation with these literals obtained through resolution.

Our cover graph-based **lgg** of RDF graphs and cover query-based **lgg** of BGPQs can be seen as the adaptation of the above techniques for clauses and θ -subsumption or standard implication to our Semantic Web setting. To this aim, we first defined the notion of anti-unification for triples and for triple patterns. Then, we established that the cover graph of two RDF graphs (resp. cover query of two BGPQs), defined as the result of all the possible anti-unifications between the triples of the input RDF graphs (resp. the triple patterns of the input BGPQs), is an **lgg** of these input RDF graphs (resp. BGPQs) w.r.t. simple entailment between RDF graphs (resp. BGPQs). Further, when the input RDF graphs (resp. BGPQs) are saturated using a set of entailment rules \mathcal{R} (resp. a set of entailment rules \mathcal{R} and a set of RDFS ontological constraints \mathcal{O}), the cover graph is an **lgg** of the input RDF graphs w.r.t. standard entailment between RDF graphs $\models_{\mathcal{R}}$ (resp. an **lgg** of the input BGPQs w.r.t. entailment between BGPQs endowed with background knowledge $\models_{\mathcal{R}, \mathcal{O}}$).

6.3 Knowledge Representation

Computing an **lgg** has started receiving consideration in the Knowledge Representation field in the early 90's, where the notion of **lgg** was rebaptized *least common subsumer* (lcs) [Cohen et al., 1992]. This problem has been mainly studied for Description Logics (DLs), and also for Conceptual Graphs (CGs).

6.3.1 Description logics

Computing an **lgg** of DL formulae, called *concepts*, has been studied for many DLs, in particular the \mathcal{EL} DL and extensions thereof [Küsters, 2001, Baader et al., 2007, Zarri  and Turhan, 2013] that shares some expressivity with RDF.

The \mathcal{EL} setting translates into *particular tree-shaped* RDF graphs, which may feature RDFS subclass and domain constraints, and for which RDF entailment is limited to the use of these two constraints only. An \mathcal{EL} concept C recursively translates into the RDF graph rooted in the blank node b_r returned by the call $\mathcal{G}(C, b_r)$, with:

- $\mathcal{G}(\top, b_r) = \emptyset$ for the universal \mathcal{EL} concept \top ,
- $\mathcal{G}(A, b_r) = \{(b_r, \tau, A)\}$ for an atomic \mathcal{EL} concept A ,

- $\mathcal{G}(\exists r.C, b_r) = \{(b_r, r, b')\} \cup \mathcal{G}(C, b')$, with b' a fresh blank node, for an \mathcal{EL} existential restriction $\exists r.C$, and
- $\mathcal{G}(C_1 \sqcap C_2, b) = \mathcal{G}(C_1, b) \cup \mathcal{G}(C_2, b)$ for an \mathcal{EL} conjunction $C_1 \sqcap C_2$.

Further, the \mathcal{EL} constraints $A_1 \sqsubseteq A_2$ and $\exists r.\top \sqsubseteq A$ correspond to (A_1, \preceq_{sc}, A_2) and (r, \leftarrow_d, A) respectively.

Example 1. Let us consider the two \mathcal{EL} concepts $C_1 \equiv \exists w.\top \sqcap \exists s.\top$ and $C_2 \equiv \exists w.\exists s.\top$ where w and s are shorthands for the `worksWith` and `hasPhdStudent` roles, respectively. C_1 represents resources/individuals working with somebody and supervising a PhD student, while C_2 denotes resources working with somebody who supervises a PhD student. The corresponding tree RDF graphs rooted in b_1 and b_2 respectively are $\mathcal{G}_1 = \mathcal{G}(C_1, b_1) = \{(b_1, w, b_{11}), (b_1, s, b_{12})\}$ and $\mathcal{G}_2 = \mathcal{G}(C_2, b_2) = \{(b_2, w, b_{21}), (b_{21}, s, b_{211})\}$ (see Figure 6.1).

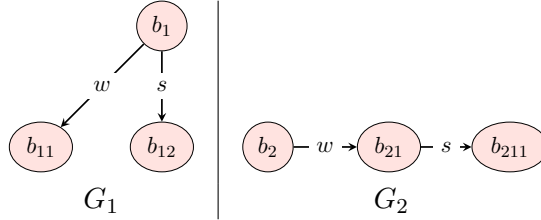


Figure 6.1 – RDF graph representation of \mathcal{EL} concepts.

In these equivalent RDF and \mathcal{EL} fragments, the \mathcal{EL} technique that computes an **lgg** of \mathcal{EL} concepts, which is an \mathcal{EL} concept, exploits the underlying tree structure of \mathcal{EL} concepts: the output concept is built by a simultaneous root-to-leaves traversal of the input concepts. We remark that it provides *only a (non least general) generalization* of their corresponding tree-shaped RDF graphs w.r.t. the problem we study, since clearly the (minimal) **lgg** of tree-shaped RDF graphs is a forest-shaped RDF graph in general as detailed in Example 2 .

Example 2. Let us consider the two tree-shaped RDF graphs \mathcal{G}_1 and \mathcal{G}_2 of Example 1. Their RDF graph **lgg** is the general forest-shaped RDF graph $\mathcal{G} = \{(b_{b_1b_2}, w, b_{b_{11}b_{21}}), (b_{b_1b_2}, s, b_{b_{12}b_{211}})\}$ (see its graphical representation displayed in Figure 6.2) while the RDF graph corresponding to their \mathcal{EL} **lgg** is the strictly more general tree-shaped RDF graph: $\mathcal{G}' = \{(b_{b_1b_2}, w, b_{b_{11}b_{21}})\}$. The latter corresponds to the \mathcal{G} subgraph rooted in $b_{b_1b_2}$, see Figure 6.2 (left), that generalizes the roots b_1 of \mathcal{G}_1 and b_2 of \mathcal{G}_2 .

Further, it can be shown that our technique for general RDF graph can be used to compute the **lgg** of \mathcal{EL} concepts as shown in Example 2 above. Roughly speaking, given two \mathcal{EL} concepts C_1, C_2 and their corresponding tree-shaped RDF graphs $\mathcal{G}(C_1, b_1), \mathcal{G}(C_2, b_2)$, the \mathcal{EL} **lgg** corresponds to the tree-shaped RDF graph rooted in b_{b_1, b_2} within the forest-shaped cover graph of $\mathcal{G}(C_1, b_1)$ and $\mathcal{G}(C_2, b_2)$.

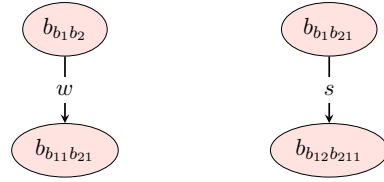


Figure 6.2 – G_{1gg} : an 1gg of RDF graphs.

6.3.2 Conceptual graphs

Computing the 1gg of CGs is briefly discussed in [Chein and Mugnier, 2009] for the so-called *simple CGs*. In particular, simple CGs with unary and binary relations correspond to *particular* RDF graphs (e.g., a property URI in a triple cannot be the subject or object of another triple, a class - URI or blank node - in a τ triple cannot be the subject of another τ triple nor the subject or object of another non- τ triple, etc.), which may feature the four RDFS constraints expressed as the *support* of the simple CGs (lattices of inclusion between unary relations and between typed binary relations), and for which entailment rules are just those exploiting these RDFS constraints [Baget et al., 2010].

The CG technique for computing an 1gg of two simple CGs, which is a simple CG, relies on a categorial product [Imrich et al., 2008] between the input graph nodes [Chein and Mugnier, 2009]. This product is the CG counterparts to all the possible anti-unifications of triples of our RDF graphs (resp. of triple patterns of our BGPQs). In contrast to our approach, and due to the simplicity of entailment in simple GCs, the input GCs do not need to be saturated w.r.t. their support; instead, required entailment are computed from the support while computing the categorial product of the input CGs.

In these equivalent RDF and CG fragments, we may interchangeably compute 1ggs with the CG technique in [Chein and Mugnier, 2009] or our for RDF graphs.

6.4 Semantic Web

Recently, the problem of computing an 1gg has started gaining interest of the Semantic Web field with the aim of finding commonalities between RDF graphs [Colucci et al., 2013, Colucci et al., 2016, El Hassad et al., 2017a] and between SPARQL queries [Lehmann and Böhmann, 2011, Böhmann et al., 2016, El Hassad et al., 2017e, El Hassad et al., 2017c].

6.4.1 RDF

In RDF, computing an 1gg has been first studied for *particular* RDF graphs, called *r-graphs* [Colucci et al., 2013, Colucci et al., 2016]. In this setting, entailment rules are

ignored: r -graphs are only compared through simple entailment (\models).

An r -graph is an *extracted subgraph* of an RDF graph \mathcal{G} , *rooted* in the \mathcal{G} value r and comprising the \mathcal{G} triples *reachable from r through directed* paths of length at most n . Such a rooted and directed r -graph can be defined recursively as $\mathcal{S}(\mathcal{G}, r, n)$, with:

- $\mathcal{S}(\mathcal{G}, r, 0) = \emptyset$
- $\mathcal{S}(\mathcal{G}, r, n) = \bigcup_{(r,p,r') \in \mathcal{G}} \{(r, p, r')\} \cup \mathcal{S}(\mathcal{G}, r', n-1) \cup \mathcal{S}(\mathcal{G}, p, n-1)$

Intuitively, this purely structural definition of r -graph attempts carrying \mathcal{G} 's knowledge about r . $\mathbf{l}gg$ s of r -graphs allow finding the commonalities between *single root entities*, while with the general RDF graphs we further allow finding the commonalities between *sets of multiple interrelated entities* [El Hassad et al., 2017a].

The technique for computing an $\mathbf{l}gg$ of two r -graphs, which is an r -graph, exploits their rooted and directed structure: it starts from their respective root and traverses them simultaneously considering triples reachable through directed paths of increasing size, while incrementally constructing an r -graph $\mathbf{l}gg$. In contrast, the general RDF graphs we consider in [El Hassad et al., 2017a] have no topological constraints. Our technique blindly traverses the input RDF graphs to anti-unify their triples with same property, and captures their common structure across these anti-unifications thanks to the consistent naming scheme we devised for the blank nodes they generate. Further, when we ignore RDF entailment, computing $\mathbf{l}gg$ s of r -graphs or of RDF graphs have the same worst-case time complexity ($O(|\mathcal{G}_1| \times |\mathcal{G}_2|)$, with $\mathcal{G}_1, \mathcal{G}_2$ the input RDF graphs). This is for instance the case for the following star-shaped RDF graphs, which are trees, r -graphs and RDF graphs, $\mathcal{G}_1 = \{(r_1, p, s_1^1), \dots, (r_1, p, s_1^m)\}$ and $\mathcal{G}_2 = \{(r_2, p, s_2^1), \dots, (r_2, p, s_2^n)\}$, the r -graph and cover graph-based $\mathbf{l}gg$ of which is built at some point by both techniques is: $\mathcal{G}_{\mathbf{l}gg} = \{(b_{r_1 r_2}, p, b_{s_1^1 s_2^1}), \dots, (b_{r_1 r_2}, p, b_{s_1^m s_2^n})\}$.

Moreover, as noted in [Colucci et al., 2016], the computed r -graphs $\mathbf{l}gg$ s are *only (non least general) generalizations* of r -graphs w.r.t. the standard semantics of RDF graphs defined upon entailment between RDF graphs using entailment rules ($\models_{\mathcal{R}}$). For instance, recall Example 2 above, in which tree-shaped RDF graphs have a forest-shaped RDF graph $\mathbf{l}gg$. Tree-shaped RDF graphs are particular r -graphs, while their forest-shaped $\mathbf{l}gg$ is not.

Therefore, our contribution in Chapter 2 significantly advance the state of the art by considering (i) *general RDF graphs* (i.e., without imposing structural restrictions) and (ii) *the standard semantics of RDF graphs based on entailment* (i.e., we take into account any set of entailment rules from the RDF standard).

6.4.2 SPARQL

Computing an $\mathbf{l}gg$ has been investigated in SPARQL for particular Basic Graph Pattern queries (BGPQs), called *unary tree-shaped BGPQ (UT-BGPQ)* [Lehmann and Böhmann, 2011, Böhmann et al., 2016], whose single answer variable is the root of its tree-shaped body.

For example, let us consider the graphs \mathcal{G}_1 and \mathcal{G}_2 in Figure 6.1. Clearly they are tree-shaped graphs and can be considered as graph representations of the two UTBGP

queries $q_1(b_1) \leftarrow (b_1, w, b_{11}), (b_1, s, b_{12})$ and $q_2(b_2) \leftarrow (b_2, w, b_{21}), (b_{21}, s, b_{211})$ whose answer variables are b_1 and b_2 respectively.

Further, in this setting, entailment rules are ignored: query are only compared through simple entailment (\models).

An **lgg** of UTBGPQs, which is a UTBGPQ, is computed by a simultaneous root-to-leaves traversal of the input queries and *returns a UTBGPQ*. This technique yields *only a (non least general) generalization* of their corresponding tree-shaped queries w.r.t. the problem we study in [El Hassad et al., 2017c]. The minimal cover query-based **lgg** of UTBGPQ is clearly a forest-shaped BGPQ. Similarly as for the \mathcal{EL} description logic above, it can be shown that our technique for computing **lgg** of BGPQ can be used to compute the UTCQ **lgg** of UTCQs as shown in Example 3.

Example 3. *Let us consider the two queries whose bodies are the tree-shaped RDF graphs \mathcal{G}_1 and \mathcal{G}_2 of Example 1. $Q_1(x_1) \leftarrow \{(x_1, w, x_{11}), (x_1, s, x_{12})\}$ and $Q_2(x_2) \leftarrow \{(x_2, w, x_{21}), (x_{21}, s, x_{211})\}$. Their **lgg** is the forest-shaped BGPQ $Q_{lgg}(x) \leftarrow \{(x, w, y), (z, s, t)\}$ while their UTBGPQ **lgg** is the strictly more general tree-shaped RDF graph: $Q'(x) \leftarrow \{(x, w, y)\}$.*

Therefore, our contribution in Chapter 3 significantly advances the state of the art on computing an **lgg** of BGP queries, by considering (i) *general BGPQs* and (ii) *any set of entailment rules and possibly extra background knowledge*.

6.5 Conclusion

In this chapter, we pointed out that the contributions of this thesis build on old ideas developed in ILP to compute then **lgg** of first order clause [Plotkin, 1970, Plotkin, 1971, Buntine, 1988, Nienhuys-Cheng and de Wolf, 1996, Nienhuys-Cheng and de Wolf, 1997].

Also, we highlighted that in the Semantic Web field, the contributions of this thesis significantly advance the state of the art by considering RDF graphs and BGPQs in all their generality, i.e., we do not impose structural restrictions on them, nor on their semantics: we use standard entailment between RDF graphs ($\models_{\mathcal{R}}$) to compute an **lgg** of RDF graphs and a well-founded extension of standard entailment between BGPQs endowed with background knowledge ($\models_{\mathcal{R}, \mathcal{O}}$) to compute an **lgg** of BGPQs.

Chapter 7

Conclusion and Perspectives

Contents

7.1 Conclusion	79
7.2 Perspectives	80
7.2.1 Redundancy elimination	80
7.2.2 Learning commonalities in DL-Lite	80

7.1 Conclusion

The goal of this thesis was to investigate the problem of finding the commonalities between descriptions of data and knowledge, formalized as their least general generalization (**lgg**), a well-known reasoning task in Machine Learning, in the setting of the popular W3C Semantic Web standards: the RDF data model and its associated SPARQL query language (Chapter 1).

The main contributions of this thesis involve the definition, characterization and computation of an **lgg** of RDF graphs (Chapter 2) and of conjunctive SPARQL queries (Chapter 3), a.k.a. Basic Graph Pattern queries (BGPQs). Crucially, our results are faithful to the RDF and SPARQL standards: by contrast to the literature, we do not restrict the structure nor semantics of RDF graphs and BGPQs (Chapter 6). Further, for BGPQs which do not carry background knowledge, we devised a well-founded generalization of standard entailment between BGPQs, in order to compute more precise **lgg**s when background knowledge is available as RDFS ontological constraints, i.e., as it is expressed within RDF graphs.

Other contributions of the thesis include a set of algorithms that allows computing an **lgg** of RDF graphs and of BGPQs based on our main technical contributions (Chapter 4). In particular, in contrast to BGPQs, RDF graphs may not fit in memory. We therefore provide algorithms for small-to-huge RDF graphs, i.e., that fit in memory, in a data management systems, and in a MapReduce cluster.

Finally, this thesis provides an experimental assessment of our technical and algorithmic contributions using synthetic LUBM data and real DBpedia data (Chapter 5).

Our experiments rely on the definition of two metrics in order to measure the gain in precision that taking into account entailment rules yields when learning an **lgg** of RDF graphs, and both entailment rules and background knowledge yield when learning an **lgg** of BGPQs.

7.2 Perspectives

7.2.1 Redundancy elimination

As a short-term perspective, we plan defining heuristics in order to efficiently prune out as much as possible redundant triples from our cover graph/query-based **lggs**, while computing them. Indeed, as for instance Figure 2.2 (Chapter 2, page 33) and Figures 3.5 (Chapter 3, pages 46) show, our technique produces many redundant triples. Such practical redundancy elimination may limit the a posteriori effort using techniques from the literature, e.g., [Pichler et al., 2013, Pichler et al., 2010, Meier, 2008].

Removing redundancy within **lggs** of RDF graphs would allow having more compact and readable **lggs**; removing redundancy within **lggs** of BGPQs may also significantly improve their evaluation time.

7.2.2 Learning commonalities in DL-Lite

As a midterm perspective, we plan to study the problem of learning **lggs** in the setting of the *DL-Lite_R* Descriptions Logics (DLs) [Calvanese et al., 2007], a first order language which underpins the OWL2 QL profile of the *Web Ontology Language*, the other Semantic Web data model by W3C, which can also be queried using SPARQL.

In this setting, we want to define, characterize and compute the commonalities between *DL-Lite_R* knowledge bases (KBs), i.e., an **lgg** of them, as well as of BGPQs when background knowledge is available as *DL-Lite_R* ontological constraints.

However, adapting the cover graph and cover query-based techniques devised in this thesis from our RDF/SPARQL setting to the *DL-Lite_R*/SPARQL setting is not obvious (at least to us), and currently open. Indeed, in this thesis, the semantics of input RDF graphs and of BGPQs is taken into account through their saturations that materializes their *finite* semantics. Unfortunately, the saturation (a.k.a. *chase*) of a *DL-Lite_R* KB, as well as that of a BGPQ w.r.t. *DL-Lite_R* ontological constraints, is *infinite* in general as the next example shows.

Example 4. Consider the *DL-Lite_R* KB made of the single fact stating that *a* is a person

$$Person(a)$$

and the constraints (here written in first order logic to facilitate the understanding to the non-expert) stating that a person has a parent, and that a parent is a person:

$$\forall x(Person(x) \Rightarrow \exists y parent(x, y)) \quad \forall x(\exists y parent(y, x) \Rightarrow Person(x))$$

Clearly, the saturation of this KB is infinite because it admits as logical consequences:

$$Person(a), parent(a, y_1), Person(y_1), parent(y_1, y_2), Parent(y_2), \dots$$

where the $y_{1 \leq i \leq \infty}$'s are existential variables.

Appendix

Appendix A

DBpedia graphs

<p>G_1 :</p> <p>"http://dbpedia.org/resource/Airbus" "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"</p> <p>"http://dbpedia.org/ontology/Organisation",</p> <p>"http://dbpedia.org/resource/Airbus_Group" "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"</p> <p>"http://dbpedia.org/ontology/Organisation",</p> <p>"http://dbpedia.org/resource/Airbus" "http://dbpedia.org/ontology/locationCountry"</p> <p>"http://dbpedia.org/resource/France",</p> <p>"http://dbpedia.org/resource/Airbus" "http://dbpedia.org/ontology/parentCompany"</p> <p>"http://dbpedia.org/resource/Airbus_Group".</p>
<p>G_2 :</p> <p>"http://dbpedia.org/resource/Hoover_Field" "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"</p> <p>"http://dbpedia.org/ontology/Airport",</p> <p>"http://dbpedia.org/resource/Hoover_Field" "http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation"</p> <p>"http://dbpedia.org/resource/Washington_metropolitan_area",</p> <p>"http://dbpedia.org/resource/Hoover_Field" "http://dbpedia.org/ontology/city"</p> <p>"http://dbpedia.org/resource/Washington_metropolitan_area",</p> <p>"http://dbpedia.org/resource/Hoover_Field" "http://dbpedia.org/ontology/owner"</p> <p>"http://dbpedia.org/resource/Henry_Berliner",</p> <p>"http://dbpedia.org/resource/Henry_Berliner" "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"</p> <p>"http://dbpedia.org/ontology/Person".</p>
<p>G_3 :</p> <p>"http://dbpedia.org/resource/Help!.(song)" "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"</p> <p>"http://dbpedia.org/ontology/Single",</p> <p>"http://dbpedia.org/resource/Help!.(song)" "http://dbpedia.org/ontology/musicalArtist"</p> <p>"http://dbpedia.org/resource/The_Beatles",</p> <p>"http://dbpedia.org/resource/The_Beatles"</p> <p>"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation" "http://dbpedia.org/resource/Liverpool",</p> <p>"http://dbpedia.org/resource/The_Beatles" "http://dbpedia.org/ontology/genre"</p> <p>"http://dbpedia.org/resource/Rock_music".</p>
<p>G_4 :</p> <p>"http://dbpedia.org/resource/The_Skeleton_Dance" "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"</p> <p>"http://dbpedia.org/ontology/Cartoon",</p> <p>"http://dbpedia.org/resource/The_Skeleton_Dance" "http://dbpedia.org/ontology/animation"</p> <p>"http://dbpedia.org/resource/Roy_O._Disney",</p> <p>"http://dbpedia.org/resource/The_Skeleton_Dance" "http://dbpedia.org/ontology/director"</p> <p>"http://dbpedia.org/resource/Walt_Disney",</p> <p>"http://dbpedia.org/resource/Roy_O._Disney" "http://dbpedia.org/ontology/birthPlace"</p> <p>"http://dbpedia.org/resource/Chicago",</p> <p>"http://dbpedia.org/resource/Walt_Disney" "http://dbpedia.org/ontology/birthPlace"</p> <p>"http://dbpedia.org/resource/Chicago",</p> <p>"http://dbpedia.org/resource/Roy_O._Disney" "http://dbpedia.org/ontology/predecessor"</p> <p>"http://dbpedia.org/resource/Walt_Disney".</p>

Figure A.1 – DBPEDIA graphs \ \mathcal{O} (A).

<p>G_5 :</p> <p>"http://dbpedia.org/resource/Winnie_the_Pooh_and_the_Honey_Tree"</p> <p>"http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Film",</p> <p>"http://dbpedia.org/resource/Winnie_the_Pooh_and_the_Honey_Tree" "http://dbpedia.org/ontology/starring"</p> <p>"http://dbpedia.org/resource/Bruce_Reitherman",</p> <p>"http://dbpedia.org/resource/Winnie_the_Pooh_and_the_Honey_Tree" "http://dbpedia.org/ontology/director"</p> <p>"http://dbpedia.org/resource/Wolfgang_Reitherman",</p> <p>"http://dbpedia.org/resource/Bruce_Reitherman" "http://dbpedia.org/ontology/birthPlace"</p> <p>"http://dbpedia.org/resource/Burbank,_California",</p> <p>"http://dbpedia.org/resource/Wolfgang_Reitherman" "http://dbpedia.org/ontology/deathPlace"</p> <p>"http://dbpedia.org/resource/Burbank,_California",</p> <p>"http://dbpedia.org/resource/Bruce_Reitherman" "http://dbpedia.org/ontology/parent"</p> <p>"http://dbpedia.org/resource/Wolfgang_Reitherman".</p>
<p>G_6 :</p> <p>"http://dbpedia.org/resource/The_Pink_Diamond" "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"</p> <p>"http://dbpedia.org/ontology/Film",</p> <p>"http://dbpedia.org/resource/The_Pink_Diamond" "http://dbpedia.org/ontology/starring"</p> <p>"http://dbpedia.org/resource/Xenia_Desni",</p> <p>"http://dbpedia.org/resource/Tamara_Desni" "http://dbpedia.org/ontology/deathPlace"</p> <p>"http://dbpedia.org/resource/France",</p> <p>"http://dbpedia.org/resource/Tamara_Desni" "http://dbpedia.org/ontology/parent"</p> <p>"http://dbpedia.org/resource/Xenia_Desni".</p>
<p>G_7 :</p> <p>"http://dbpedia.org/resource/Too_Much_Sun" "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"</p> <p>"http://dbpedia.org/ontology/Film",</p> <p>"http://dbpedia.org/resource/Too_Much_Sun" "http://dbpedia.org/ontology/starring"</p> <p>"http://dbpedia.org/resource/Robert_Downey,_Jr.",</p> <p>"http://dbpedia.org/resource/Too_Much_Sun" "http://dbpedia.org/ontology/director"</p> <p>"http://dbpedia.org/resource/Robert_Downey,_Sr.",</p> <p>"http://dbpedia.org/resource/Robert_Downey,_Jr." "http://dbpedia.org/ontology/birthPlace"</p> <p>"http://dbpedia.org/resource/Manhattan",</p> <p>"http://dbpedia.org/resource/Robert_Downey,_Sr." "http://dbpedia.org/ontology/birthPlace"</p> <p>"http://dbpedia.org/resource/New_York_City",</p> <p>"http://dbpedia.org/resource/Robert_Downey,_Jr." "http://dbpedia.org/ontology/parent"</p> <p>"http://dbpedia.org/resource/Robert_Downey,_Sr.".</p>
<p>G_8 :</p> <p>"http://dbpedia.org/resource/The_Boy_and_the_Pirates" "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"</p> <p>"http://dbpedia.org/ontology/Film",</p> <p>"http://dbpedia.org/resource/The_Boy_and_the_Pirates" "http://dbpedia.org/ontology/starring"</p> <p>"http://dbpedia.org/resource/Susan_Gordon",</p> <p>"http://dbpedia.org/resource/The_Boy_and_the_Pirates" "http://dbpedia.org/ontology/director"</p> <p>"http://dbpedia.org/resource/Bert_I._Gordon",</p> <p>"http://dbpedia.org/resource/Susan_Gordon" "http://dbpedia.org/ontology/deathPlace"</p> <p>"http://dbpedia.org/resource/Teaneck,_New_Jersey",</p> <p>"http://dbpedia.org/resource/Susan_Gordon" "http://dbpedia.org/ontology/parent"</p> <p>"http://dbpedia.org/resource/Bert_I._Gordon".</p>
<p>G_9 :</p> <p>"http://dbpedia.org/resource/Alexandra_Cousteau" "http://dbpedia.org/ontology/parent"</p> <p>"http://dbpedia.org/resource/Philippe_Cousteau",</p> <p>"http://dbpedia.org/resource/Philippe_Cousteau" "http://dbpedia.org/ontology/birthPlace"</p> <p>"http://dbpedia.org/resource/France",</p> <p>"http://dbpedia.org/resource/Voyage_to_the_Edge_of_the_World" "http://dbpedia.org/ontology/director"</p> <p>"http://dbpedia.org/resource/Marshall_Flaum",</p> <p>"http://dbpedia.org/resource/Voyage_to_the_Edge_of_the_World" "http://dbpedia.org/ontology/starring"</p> <p>"http://dbpedia.org/resource/Philippe_Cousteau",</p> <p>"http://dbpedia.org/resource/Voyage_to_the_Edge_of_the_World"</p> <p>"http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Film".</p>

Figure A.2 – DBPEDIA graphs \ \mathcal{O} (B).

Appendix B

LUBM queries

<p>$Q_1(?X, ?Y) :-$ $?X \text{ "http://www.w3.org/1999/02/22-rdf-syntax-ns\#type" "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#Employee",}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#worksFor" "http://www.Department0.University0.edu",}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#degreeFrom" ?Y}$</p>
<p>$Q_2(?X, ?Y) :-$ $?X \text{ "http://www.w3.org/1999/02/22-rdf-syntax-ns\#type" "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#Employee",}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#worksFor" "http://www.Department0.University0.edu",}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#doctoralDegreeFrom" ?Y}$</p>
<p>$Q_3(?X, ?Y, ?Z) :-$ $?X \text{ "http://www.w3.org/1999/02/22-rdf-syntax-ns\#type" "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#Student",}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#advisor" ?Y,}$ $?Y \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#teacherOf" ?Z,}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#takesCourse" ?Z}$</p>
<p>$Q_4(?X, ?Y) :-$ $?X \text{ "http://www.w3.org/1999/02/22-rdf-syntax-ns\#type" "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#Faculty",}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#degreeFrom" ?Y,}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#memberOf" ?Y}$</p>
<p>$Q_5(?X, ?Y) :-$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#degreeFrom" ?Y,}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#memberOf" "http://www.Department0.University0.edu",}$</p>
<p>$Q_6(?W, ?X, ?Y) :-$ $?X \text{ "http://www.w3.org/1999/02/22-rdf-syntax-ns\#type" "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#GraduateStudent",}$ $?Y \text{ "http://www.w3.org/1999/02/22-rdf-syntax-ns\#type" "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#Faculty",}$ $?W \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#publicationAuthor" ?X,}$ $?W \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#publicationAuthor" ?Y}$</p>
<p>$Q_7(?W, ?X, ?Y) :-$ $?X \text{ "http://www.w3.org/1999/02/22-rdf-syntax-ns\#type" "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#GraduateStudent",}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#advisor" ?Y,}$ $?Y \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#teacherOf" ?Z,}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#takesCourse" ?Z,}$ $?W \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#publicationAuthor" ?X,}$ $?W \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#publicationAuthor" ?Y}$</p>
<p>$Q_8(?X, ?Y) :-$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#doctoralDegreeFrom" ?Z,}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#teacherOf" ?Y}$</p>
<p>$Q_9(?X, ?Y) :-$ $?X \text{ "http://www.w3.org/1999/02/22-rdf-syntax-ns\#type" "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#Faculty",}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#degreeFrom" "http://www.University532.edu",}$ $?X \text{ "http://swat.cse.lehigh.edu/onto/univ-bench.owl\#memberOf" ?Y}$</p>

Figure B.1 – LUBM queries.

Appendix C

DBpedia queries

<p>$Q_1(?X, ?Y) : -$ $?X$ "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Organisation", $?Y$ "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Organisation", $?X$ "http://dbpedia.org/ontology/locationCountry" "http://dbpedia.org/resource/France", $?Y$ "http://dbpedia.org/ontology/parentCompany" ?Y</p>
<p>$Q_2(?X, ?Y) : -$ $?X$ "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Airport", $?X$ "http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation" "http://dbpedia.org/resource/France", $?X$ "http://dbpedia.org/ontology/city" ?Z, $?X$ "http://dbpedia.org/ontology/owner" ?Y, $?Y$ "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Person", $?V$ "http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation" ?X</p>
<p>$Q_3(?X, ?Y) : -$ $?X$ "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Single", $?X$ "http://dbpedia.org/ontology/musicalArtist" ?Y, $?Y$ "http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation" ?Z, $?Y$ "http://dbpedia.org/ontology/genre" ?V,</p>
<p>$Q_4(?X, ?Y) : -$ $?X$ "http://dbpedia.org/ontology/birthPlace" ?V, $?X$ "http://dbpedia.org/ontology/parent" ?Y, $?Y$ "http://dbpedia.org/ontology/deathPlace" ?V, $?Z$ "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Film", $?Z$ "http://dbpedia.org/ontology/starring" ?X, $?Z$ "http://dbpedia.org/ontology/director" ?Y</p>
<p>$Q_5(?X, ?Y) : -$ $?X$ "http://dbpedia.org/ontology/deathPlace" "http://dbpedia.org/resource/France", $?X$ "http://dbpedia.org/ontology/parent" ?Y, $?Z$ "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Film", $?Z$?V ?Y</p>
<p>$Q_6(?X, ?Y) : -$ $?X$ "http://dbpedia.org/ontology/birthPlace" "http://dbpedia.org/resource/France", $?Y$ "http://dbpedia.org/ontology/birthPlace" "http://dbpedia.org/resource/France", $?X$ "http://dbpedia.org/resource/predecessor" ?Y, $?Z$ "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Cartoon", $?Z$ "http://dbpedia.org/ontology/ animator" ?X, $?Z$ "http://dbpedia.org/ontology/director" ?Y</p>
<p>$Q_7(?X, ?Y) : -$ $?X$ "http://dbpedia.org/ontology/birthPlace" "http://dbpedia.org/resource/France", $?Y$ "http://dbpedia.org/ontology/birthPlace" "http://dbpedia.org/resource/France", $?X$ "http://dbpedia.org/resource/parent" ?Y, $?Z$ "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Film", $?Z$ "http://dbpedia.org/ontology/starring" ?X, $?Z$ "http://dbpedia.org/ontology/director" ?Y</p>
<p>$Q_8(?X, ?Y) : -$ $?X$ "http://dbpedia.org/ontology/deathPlace" "http://dbpedia.org/resource/England", $?Y$ "http://dbpedia.org/ontology/deathPlace" "http://dbpedia.org/resource/England", $?X$ "http://dbpedia.org/resource/parent" ?Y, $?Z$ "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "http://dbpedia.org/ontology/Film", $?Z$ "http://dbpedia.org/ontology/starring" ?X, $?Z$ "http://dbpedia.org/ontology/director" ?Y</p>

Figure C.1 – DBPEDIA queries.

Appendix D

DBpedia extracted ontology \mathcal{O}

```

"http://dbpedia.org/ontology/Language" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.w3.org/2002/07/owl#Thing",
"http://dbpedia.org/ontology/BusinessPerson" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/sourceConfluenceState" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/sourceConfluenceState" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/sourceConfluenceState" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/SportsClub" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/Criminal" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Journalist" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/MilitaryUnit" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/map" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/map" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Group" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://schema.org/Organization",
"http://dbpedia.org/ontology/Group" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/GovernmentAgency" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/NaturalPlace" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/Orphan" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Artwork" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Work",
"http://dbpedia.org/ontology/chiefPlace" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/chiefPlace" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Place" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.w3.org/2002/07/owl#Thing",
"http://dbpedia.org/ontology/Engineer" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Politician" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/sourceConfluencePosition" "http://www.w3.org/2000/01/rdf-schema#range"
"http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing",
"http://dbpedia.org/ontology/sourceConfluencePosition" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/sourceConfluencePosition" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Coach" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/mouthState" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/mouthState" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/mouthState" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/spokenIn" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/spokenIn" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Language",

```

Figure D.1 – DBPEDIA extracted ontology \mathcal{O} (1).

```

"http://dbpedia.org/ontology/spokenIn" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/homeport" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/homeport" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Ship",
"http://dbpedia.org/ontology/homeport" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/parent" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#sameSettingAs",
"http://dbpedia.org/ontology/parent" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/parent" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Writer" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/genre" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isClassifiedBy",
"http://dbpedia.org/ontology/genre" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Genre",
"http://dbpedia.org/ontology/Non-ProfitOrganisation" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/HorseTrainer" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Spacecraft" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/MeansOfTransportation",
"http://dbpedia.org/ontology/Athlete" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Biomolecule" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.w3.org/2002/07/owl#Thing",
"http://dbpedia.org/ontology/TopicalConcept" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.w3.org/2002/07/owl#Thing",
"http://dbpedia.org/ontology/county" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/county" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/massif" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/massif" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/SkiResort",
"http://dbpedia.org/ontology/massif" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Producer" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/PublicTransitSystem" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/EducationalInstitution" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/SkiArea" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/SportFacility",
"http://dbpedia.org/ontology/Presenter" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Island" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/capitalDistrict" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/capitalDistrict" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Island",
"http://dbpedia.org/ontology/capitalDistrict" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/mouthPlace" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",

```

Figure D.2 – DBPEDIA extracted ontology \mathcal{O} (2).

```

"http://dbpedia.org/ontology/mouthPlace" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/mouthPlace" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Animator" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#coparticipatesWith",
"http://dbpedia.org/ontology/Animator" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Agent",
"http://dbpedia.org/ontology/Animator" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Cartoon",
"http://dbpedia.org/ontology/SiteOfSpecialScientificInterest" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/BodyOfWater" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/NaturalPlace",
"http://dbpedia.org/ontology/arrondissement" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/arrondissement" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/arrondissement" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/ArchitecturalStructure" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/sourceMountain" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Mountain",
"http://dbpedia.org/ontology/sourceMountain" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/hubAirport" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Airport",
"http://dbpedia.org/ontology/hubAirport" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Airline",
"http://dbpedia.org/ontology/hubAirport" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/ruralMunicipality" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/ruralMunicipality" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Road",
"http://dbpedia.org/ontology/ruralMunicipality" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/areaOfSearch" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/areaOfSearch" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/SiteOfSpecialScientificInterest",
"http://dbpedia.org/ontology/areaOfSearch" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Aristocrat" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/TelevisionDirector" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Museum" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Building",
"http://dbpedia.org/ontology/capitalCountry" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Country",
"http://dbpedia.org/ontology/capitalCountry" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Island",
"http://dbpedia.org/ontology/capitalCountry" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/EthnicGroup" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.w3.org/2002/07/owl#Thing",
"http://dbpedia.org/ontology/City" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Settlement",
"http://dbpedia.org/ontology/canton" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Settlement",

```

Figure D.3 – DBPEDIA extracted ontology \mathcal{O} (3).

```

"http://dbpedia.org/ontology/canton" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Settlement",
"http://dbpedia.org/ontology/canton" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Eukaryote" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Species",
"http://dbpedia.org/ontology/campus" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/University",
"http://dbpedia.org/ontology/campus" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/placeOfBurial" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/placeOfBurial" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/placeOfBurial" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/sourceCountry" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Country",
"http://dbpedia.org/ontology/sourceCountry" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Stream",
"http://dbpedia.org/ontology/sourceCountry" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Organisation" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Agent",
"http://dbpedia.org/ontology/Road" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/RouteOfTransportation",
"http://dbpedia.org/ontology/owningOrganisation" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/owningOrganisation" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://dbpedia.org/ontology/owner",
"http://dbpedia.org/ontology/Bridge" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/RouteOfTransportation",
"http://dbpedia.org/ontology/mouthDistrict" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/mouthDistrict" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/mouthDistrict" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/place" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/place" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/MilitaryConflict",
"http://dbpedia.org/ontology/place" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/TheatreDirector" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Country" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/geneLocation" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/geneLocation",
"http://dbpedia.org/ontology/geneLocation" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/gene",
"http://dbpedia.org/ontology/geneLocation" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/broadcastArea" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/broadcastArea" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Broadcaster",
"http://dbpedia.org/ontology/broadcastArea" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/PoliticianSpouse" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",

```

Figure D.4 – DBPEDIA extracted ontology \mathcal{O} (4).


```

"http://dbpedia.org/ontology/alpsSupergroup" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/MountainRange",
"http://dbpedia.org/ontology/alpsSupergroup" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Mountain",
"http://dbpedia.org/ontology/alpsSupergroup" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Species" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.w3.org/2002/07/owl#Thing",
"http://dbpedia.org/ontology/sourcePosition" "http://www.w3.org/2000/01/rdf-schema#range"
"http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing",
"http://dbpedia.org/ontology/sourcePosition" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/alpsSection" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/MountainRange",
"http://dbpedia.org/ontology/alpsSection" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Mountain",
"http://dbpedia.org/ontology/alpsSection" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/settlement" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/settlement" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/settlement" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/SkiResort" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/SkiArea",
"http://dbpedia.org/ontology/Parliament" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/sourcePlace" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/sourcePlace" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/TradeUnion" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/Economist" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/predecessor" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#sameSettingAs",
"http://dbpedia.org/ontology/sourceDistrict" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/sourceDistrict" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/nationality" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Country",
"http://dbpedia.org/ontology/nationality" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/nationality" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Work" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.w3.org/2002/07/owl#Thing",
"http://dbpedia.org/ontology/MeanOfTransportation" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.w3.org/2002/07/owl#Thing",
"http://dbpedia.org/ontology/Psychologist" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/MountainRange" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/NaturalPlace",
"http://dbpedia.org/ontology/birthPlace" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/birthPlace" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/birthPlace" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",

```

Figure D.5 – DBPEDIA extracted ontology \mathcal{O} (5).

```

"http://dbpedia.org/ontology/linkedTo" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/linkedTo" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/SkiResort",
"http://dbpedia.org/ontology/linkedTo" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/headquarter" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/headquarter" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/headquarter" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/MilitaryPerson" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/WineRegion" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/Model" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/TermOfOffice" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/borough" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/borough" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/borough" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/bodyDiscovered" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/bodyDiscovered" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/bodyDiscovered" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Monarch" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/targetAirport" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Airport",
"http://dbpedia.org/ontology/targetAirport" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Airline",
"http://dbpedia.org/ontology/targetAirport" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/museum" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Museum",
"http://dbpedia.org/ontology/museum" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Artwork",
"http://dbpedia.org/ontology/museum" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Airline" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Company",
"http://dbpedia.org/ontology/populationPlace" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/populationPlace" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/EthnicGroup",
"http://dbpedia.org/ontology/populationPlace" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/OrganisationMember" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/PopulatedPlace" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/Royalty" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/alpsSubgroup" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/MountainRange",

```

Figure D.6 – DBPEDIA extracted ontology \mathcal{O} (6).

```

"http://dbpedia.org/ontology/alpsSubgroup" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Mountain",
"http://dbpedia.org/ontology/alpsSubgroup" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/garrison" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/garrison" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/MilitaryUnit",
"http://dbpedia.org/ontology/garrison" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/hometown" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Settlement",
"http://dbpedia.org/ontology/hometown" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Agent",
"http://dbpedia.org/ontology/hometown" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/director" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#coparticipatesWith",
"http://dbpedia.org/ontology/director" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/director" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Film",
"http://dbpedia.org/ontology/SportsTeam" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/SocietalEvent" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Event",
"http://dbpedia.org/ontology/europeanAffiliation" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/PoliticalParty",
"http://dbpedia.org/ontology/europeanAffiliation" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/meetingBuilding" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Building",
"http://dbpedia.org/ontology/meetingBuilding" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Legislature",
"http://dbpedia.org/ontology/meetingBuilding" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/MusicalArtist" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#NaturalPerson",
"http://dbpedia.org/ontology/MusicalArtist" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://schema.org/MusicGroup",
"http://dbpedia.org/ontology/MusicalArtist" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Artist",
"http://dbpedia.org/ontology/Event" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.w3.org/2002/07/owl#Thing",
"http://dbpedia.org/ontology/OfficeHolder" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/EmployersOrganisation" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/Noble" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/sourceRegion" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/sourceRegion" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/literaryGenre" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#isClassifiedBy",
"http://dbpedia.org/ontology/literaryGenre" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/WrittenWork",
"http://dbpedia.org/ontology/literaryGenre" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://dbpedia.org/ontology/genre",
"http://dbpedia.org/ontology/PoliticalParty" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",

```

Figure D.7 – DBPEDIA extracted ontology \mathcal{O} (7).

```

"http://dbpedia.org/ontology/Building" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/ArchitecturalStructure",
"http://dbpedia.org/ontology/InternationalOrganisation" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/sourceConfluenceCountry" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Country",
"http://dbpedia.org/ontology/sourceConfluenceCountry" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/sourceConfluenceCountry" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/locationCity" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://dbpedia.org/ontology/location",
"http://dbpedia.org/ontology/locationCity" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/City",
"http://dbpedia.org/ontology/locationCity" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/locationCity" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Grape" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/FloweringPlant",
"http://dbpedia.org/ontology/SpaceStation" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/MeanOfTransportation",
"http://dbpedia.org/ontology/capitalMountain" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Mountain",
"http://dbpedia.org/ontology/capitalMountain" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Island",
"http://dbpedia.org/ontology/capitalMountain" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/wineRegion" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/WineRegion",
"http://dbpedia.org/ontology/wineRegion" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Grape",
"http://dbpedia.org/ontology/wineRegion" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/beltwayCity" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/City",
"http://dbpedia.org/ontology/beltwayCity" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Road",
"http://dbpedia.org/ontology/beltwayCity" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/capitalRegion" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/capitalRegion" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Island",
"http://dbpedia.org/ontology/capitalRegion" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Airport" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Infrastructure",
"http://dbpedia.org/ontology/Agent" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.w3.org/2002/07/owl#Thing",
"http://dbpedia.org/ontology/SportsLeague" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/wilaya" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/wilaya" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Settlement",
"http://dbpedia.org/ontology/wilaya" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/owner" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#sameSettingAs",
"http://dbpedia.org/ontology/owner" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Agent",

```

Figure D.8 – DBPEDIA extracted ontology \mathcal{O} (8).

```

"http://dbpedia.org/ontology/Film" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Work",
"http://dbpedia.org/ontology/ReligiousOrganisation" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/Broadcaster" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/starring" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#sameSettingAs",
"http://dbpedia.org/ontology/starring" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Actor",
"http://dbpedia.org/ontology/starring" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Work",
"http://dbpedia.org/ontology/Philosopher" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/locatedInArea" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/locatedInArea" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/locatedInArea" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/alpsSubsection" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/MountainRange",
"http://dbpedia.org/ontology/alpsSubsection" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Mountain",
"http://dbpedia.org/ontology/alpsSubsection" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/capitalPosition" "http://www.w3.org/2000/01/rdf-schema#range"
"http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing",
"http://dbpedia.org/ontology/capitalPosition" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Island",
"http://dbpedia.org/ontology/capitalPosition" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/sourceConfluencePlace" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/sourceConfluencePlace" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/sourceConfluencePlace" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/SportsManager" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Archeologist" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/state" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/state" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Infrastructure" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/ArchitecturalStructure",
"http://dbpedia.org/ontology/mouthMountain" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Mountain",
"http://dbpedia.org/ontology/mouthMountain" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/mouthMountain" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Company" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/BeautyQueen" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Plant" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Eukaryote",
"http://dbpedia.org/ontology/mouthRegion" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",

```

Figure D.9 – DBPEDIA extracted ontology \mathcal{O} (9).

```

"http://dbpedia.org/ontology/mouthRegion" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/mouthRegion" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Settlement" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/Celebrity" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/sourceState" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/sourceState" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/MilitaryConflict" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/SocietalEvent",
"http://dbpedia.org/ontology/musicalArtist" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#coparticipatesWith",
"http://dbpedia.org/ontology/musicalArtist" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/MusicalArtist",
"http://dbpedia.org/ontology/musicalArtist" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Single",
"http://dbpedia.org/ontology/Mountain" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/NaturalPlace",
"http://dbpedia.org/ontology/SambaSchool" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/Legislature" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/region" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/region" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Judge" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Gene" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Biomolecule",
"http://dbpedia.org/ontology/meetingCity" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Settlement",
"http://dbpedia.org/ontology/meetingCity" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Legislature",
"http://dbpedia.org/ontology/meetingCity" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/restingPlace" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/restingPlace" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/restingPlace" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/residence" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/residence" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/residence" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/mouthPosition" "http://www.w3.org/2000/01/rdf-schema#range"
"http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing",
"http://dbpedia.org/ontology/mouthPosition" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/mouthPosition" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/countySeat" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/countySeat" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",

```

Figure D.10 – DBPEDIA extracted ontology \mathcal{O} (10).

```

"http://dbpedia.org/ontology/TelevisionPersonality" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/WrittenWork" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Work",
"http://dbpedia.org/ontology/alpsMainPart" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/MountainRange",
"http://dbpedia.org/ontology/alpsMainPart" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Mountain",
"http://dbpedia.org/ontology/alpsMainPart" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/FloweringPlant" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Plant",
"http://dbpedia.org/ontology/Ship" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/MeanOfTransportation",
"http://dbpedia.org/ontology/Scientist" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Stream" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/BodyOfWater",
"http://dbpedia.org/ontology/nationalAffiliation" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/PoliticalParty",
"http://dbpedia.org/ontology/nationalAffiliation" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Single" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/MusicalWork",
"http://dbpedia.org/ontology/Referee" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/country" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Country",
"http://dbpedia.org/ontology/country" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/GeneLocation" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://www.w3.org/2002/07/owl#Thing",
"http://dbpedia.org/ontology/FictionalCharacter" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Person" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Agent",
"http://dbpedia.org/ontology/Religious" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Chef" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/crosses" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/crosses" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Bridge",
"http://dbpedia.org/ontology/crosses" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Cleric" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Actor" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Artist",
"http://dbpedia.org/ontology/MusicalWork" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Work",
"http://dbpedia.org/ontology/deathPlace" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/deathPlace" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/deathPlace" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/RouteOfTransportation" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Infrastructure",
"http://dbpedia.org/ontology/location" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",

```

Figure D.11 – DBPEDIA extracted ontology \mathcal{O} (11).

```

"http://dbpedia.org/ontology/location" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/stateOfOrigin" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Country",
"http://dbpedia.org/ontology/stateOfOrigin" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/stateOfOrigin" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/daira" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/daira" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Settlement",
"http://dbpedia.org/ontology/daira" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Astronaut" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/ Cartoon" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Work",
"http://dbpedia.org/ontology/Farmer" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/parentCompany" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#sameSettingAs",
"http://dbpedia.org/ontology/parentCompany" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Company",
"http://dbpedia.org/ontology/University" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/EducationalInstitution",
"http://dbpedia.org/ontology/GeopoliticalOrganisation" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/SportFacility" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/ArchitecturalStructure",
"http://dbpedia.org/ontology/Artist" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/RomanEmperor" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Genre" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/TopicalConcept",
"http://dbpedia.org/ontology/restingPlacePosition" "http://www.w3.org/2000/01/rdf-schema#range"
"http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing",
"http://dbpedia.org/ontology/restingPlacePosition" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/restingPlacePosition" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/capitalPlace" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/PopulatedPlace",
"http://dbpedia.org/ontology/capitalPlace" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Island",
"http://dbpedia.org/ontology/capitalPlace" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Ambassador" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/foundationPlace" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/City",
"http://dbpedia.org/ontology/foundationPlace" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Organisation",
"http://dbpedia.org/ontology/foundationPlace" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Linguist" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/alpsMajorSector" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/MountainRange",
"http://dbpedia.org/ontology/alpsMajorSector" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Mountain",

```

Figure D.12 – DBPEDIA extracted ontology \mathcal{O} (12).


```

"http://dbpedia.org/ontology/alpsMajorSector" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/MovieDirector" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/mouthCountry" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Country",
"http://dbpedia.org/ontology/mouthCountry" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/mouthCountry" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/targetSpaceStation" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/SpaceStation",
"http://dbpedia.org/ontology/targetSpaceStation" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Spacecraft",
"http://dbpedia.org/ontology/targetSpaceStation" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/Lawyer" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/River" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Stream",
"http://dbpedia.org/ontology/locationCountry" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/locationCountry" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://dbpedia.org/ontology/location",
"http://dbpedia.org/ontology/locationCountry" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Country",
"http://dbpedia.org/ontology/MemberResistanceMovement" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Architect" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/sourceConfluenceRegion" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/Place",
"http://dbpedia.org/ontology/sourceConfluenceRegion" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/River",
"http://dbpedia.org/ontology/sourceConfluenceRegion" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/alpsGroup" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/MountainRange",
"http://dbpedia.org/ontology/alpsGroup" "http://www.w3.org/2000/01/rdf-schema#domain"
"http://dbpedia.org/ontology/Mountain",
"http://dbpedia.org/ontology/alpsGroup" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation",
"http://dbpedia.org/ontology/PlayboyPlaymate" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/Egyptologist" "http://www.w3.org/2000/01/rdf-schema#subClassOf"
"http://dbpedia.org/ontology/Person",
"http://dbpedia.org/ontology/city" "http://www.w3.org/2000/01/rdf-schema#range"
"http://dbpedia.org/ontology/City",
"http://dbpedia.org/ontology/city" "http://www.w3.org/2000/01/rdf-schema#subPropertyOf"
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation".

```

Figure D.13 – DBPEDIA extracted ontology \mathcal{O} (13).

Bibliography

- [db2, 2012] (2012). DB2. www.ibm.com/analytics/us/en/technology/db2. Online; last access 23 November 2017.
- [had, 2016] (2016). Hadoop. hadoop.apache.org.
- [pos, 2016] (2016). PostgreSQL. www.postgresql.org. Online; last access 23 November 2017.
- [jen, 2017] (2017). Jena. jena.apache.org. Online; last access 23 November 2017.
- [mys, 2017] (2017). MySQL. www.mysql.com. Online; last access 23 November 2017.
- [ora, 2017] (2017). Oracle. www.oracle.com/database. Online; last access 23 November 2017.
- [spa, 2017] (2017). Spark. spark.apache.org.
- [vir, 2017] (2017). Virtuoso. virtuoso.openlinksw.com. Online; last access 23 November 2017.
- [Baader et al., 1999] Baader, F., Kiisters, R., and Molitor, R. (1999). Computing least common subsumers in description logics with existential restrictions. In *IJCAI*.
- [Baader et al., 2007] Baader, F., Sertkaya, B., and Turhan, A.-Y. (2007). Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logic*, 5(3).
- [Baget et al., 2010] Baget, J., Croitoru, M., Gutierrez, A., Leclère, M., and Mugnier, M. (2010). Translations between RDF(S) and conceptual graphs. In *ICCS*.
- [Bornea et al., 2013] Bornea, M. A., Dolby, J., Kementsietsidis, A., Srinivas, K., Dantressangle, P., Udrea, O., and Bhattacharjee, B. (2013). Building an efficient RDF store over a relational database. In *SIGMOD*.
- [Bühmann et al., 2016] Bühmann, L., Lehmann, J., and Westphal, P. (2016). DL-Learner - a framework for inductive learning on the semantic web. *J. Web Semantics*, 39.
- [Buntine, 1988] Buntine, W. L. (1988). Generalized subsumption and its applications to induction and redundancy. *Artif. Intell.*, 36(2):149–176.
- [Bursztyn et al., 2015] Bursztyn, D., Goasdoué, F., and Manolescu, I. (2015). Optimizing reformulation-based query answering in RDF. In *EDBT*.
- [Bursztyn et al., 2016] Bursztyn, D., Goasdoué, F., and Manolescu, I. (2016). Teaching an RDBMS about ontological constraints. *PVLDB*, 9(12).

- [Calvanese et al., 2007] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. (2007). Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated Reasoning*, 39(3):385–429.
- [Chein and Mugnier, 2009] Chein, M. and Mugnier, M. (2009). *Graph-based Knowledge Representation - Computational Foundations of Conceptual Graphs*. Springer.
- [Cheng et al., 2014] Cheng, G., Zhang, Y., and Qu, Y. (2014). Explass: Exploring associations between entities via top-k ontological patterns and facets. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*, pages 422–437.
- [Chuang and Chien, 2002] Chuang, S.-L. and Chien, L.-F. (2002). Towards automatic generation of query taxonomy: A hierarchical query clustering approach. In *ICDM*.
- [Cohen et al., 1992] Cohen, W. W., Borgida, A., and Hirsh, H. (1992). Computing least common subsumers in description logics. In *AAAI*.
- [Colazzo et al., 2014] Colazzo, D., Goasdoué, F., Manolescu, I., and Roatis, A. (2014). RDF analytics: lenses over semantic graphs. In *WWW*.
- [Colucci et al., 2016] Colucci, S., Donini, F., Giannini, S., and Sciascio, E. D. (2016). Defining and computing least common subsumers in RDF. *J. Web Semantics*, 39(0).
- [Colucci et al., 2013] Colucci, S., Donini, F. M., and Sciascio, E. D. (2013). Common subsumers in RDF. In *AI*IA*.
- [Dean and Ghemawat, 2004] Dean, J. and Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. In *OSDI*.
- [El Hassad et al., 2017a] El Hassad, S., Goasdoué, F., and Jaudoin, H. (2017a). Learning commonalities in RDF. In *The 14th Extended Semantic Web Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*, pages 502–517.
- [El Hassad et al., 2017b] El Hassad, S., Goasdoué, F., and Jaudoin, H. (2017b). Learning commonalities in RDF. In *Already published paper track, 27th International Conference on Inductive Logic Programming (ILP 2017), Orléans, France, September 4 - September 6, 2017*.
- [El Hassad et al., 2017c] El Hassad, S., Goasdoué, F., and Jaudoin, H. (2017c). Learning commonalities in SPARQL. In *The 16th International Semantic Web Conference, ISWC 2017, Vienna, Austria, October 21 - October 25, 2017*.
- [El Hassad et al., 2017d] El Hassad, S., Goasdoué, F., and Jaudoin, H. (2017d). Learning commonalities in SPARQL. In *Already published paper track, (BDA 2017), Nancy, France, November 14-17*.
- [El Hassad et al., 2017e] El Hassad, S., Goasdoué, F., and Jaudoin, H. (2017e). Towards learning commonalities in SPARQL. In *ESWC*. poster track.
- [Garcia-Molina et al., 2009] Garcia-Molina, H., Ullman, J. D., and Widom, J. (2009). *Database systems - the complete book*. Pearson Education.
- [Goasdoué et al., 2015] Goasdoué, F., Kaoudi, Z., Manolescu, I., Quiané-Ruiz, J., and Zampetakis, S. (2015). Cliquesquare: Flat plans for massively parallel RDF queries. In *ICDE*.

- [Goasdoué et al., 2011] Goasdoué, F., Karanasos, K., Leblay, J., and Manolescu, I. (2011). View selection in semantic web databases. *PVLDB*, 5(2).
- [Goasdoué et al., 2013] Goasdoué, F., Manolescu, I., and Roatis, A. (2013). Efficient query answering against dynamic RDF databases. In *EDBT*.
- [Grimnes et al., 2008] Grimnes, G. A., Edwards, P., and Preece, A. D. (2008). Instance based clustering of semantic web resources. In *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, pages 303–317.
- [Guo et al., 2005] Guo, Y., Pan, Z., and Heflin, J. (2005). LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3).
- [Heckel et al., 2017] Heckel, R., Vlachos, M., Parnell, T. P., and Dünner, C. (2017). Scalable and interpretable product recommendations via overlapping co-clustering. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 1033–1044.
- [Huang et al., 2016] Huang, Z., Cautis, B., Cheng, R., and Zheng, Y. (2016). Kb-enabled query recommendation for long-tail queries. In *CIKM*.
- [Husain et al., 2011] Husain, M. F., McGlothlin, J. P., Masud, M. M., Khan, L. R., and Thuraisingham, B. M. (2011). Heuristics-based query processing for large RDF graphs using cloud computing. *IEEE TKDE*, 23(9).
- [Imrich et al., 2008] Imrich, W., Klavzar, S., and Rall, D. F. (2008). *Topics in Graph Theory: Graphs and Their Cartesian Product*. AK Peters Ltd.
- [Jin et al., 2005] Jin, R., Wang, C., Polshakov, D., Parthasarathy, S., and Agrawal, G. (2005). Discovering frequent topological structures from graph datasets. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 606–611.
- [Küsters, 2001] Küsters, R. (2001). *Non-Standard Inferences in Description Logics*, volume 2100 of *LNCS*. Springer.
- [Kuzelka et al., 2012] Kuzelka, O., Szabóová, A., and Zelezný, F. (2012). Bounded least general generalization. In *Inductive Logic Programming - 22nd International Conference, ILP 2012, Dubrovnik, Croatia, September 17-19, 2012, Revised Selected Papers*, pages 116–129.
- [Le et al., 2012] Le, W., Kementsietsidis, A., Duan, S., and Li, F. (2012). Scalable multi-query optimization for SPARQL. In *ICDE*.
- [Lee and Liu, 2013] Lee, K. and Liu, L. (2013). Scaling queries over big RDF graphs with semantic hash partitioning. *PVLDB*, 6(14).
- [Lehmann and Bühmann, 2011] Lehmann, J. and Bühmann, L. (2011). Autosparql: Let users query your knowledge base. In *ESWC*.
- [Lehmann et al., 2015] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2015). DBpedia. *Semantic Web*, 6(2):167–195.

- [Lisi, 2007] Lisi, F. (2007). The challenges of the semantic web to machine learning and data mining. tutorial @ ecml/pkdd 2007.
- [Lohmann et al., 2010] Lohmann, S., Heim, P., Stegemann, T., and Ziegler, J. (2010). The refinder user interface: interactive exploration of relationships between objects of interest. In *Proceedings of the 15th International Conference on Intelligent User Interfaces, IUI 2010, Hong Kong, China, February 7-10, 2010*, pages 421–422.
- [Meier, 2008] Meier, M. (2008). Towards rule-based minimization of RDF graphs under constraints. In *RR*.
- [Neumann and Weikum, 2010a] Neumann, T. and Weikum, G. (2010a). The RDF-3X engine for scalable management of RDF data. *VLDB J.*, 19(1).
- [Neumann and Weikum, 2010b] Neumann, T. and Weikum, G. (2010b). x-rdf-3x: Fast querying, high update rates, and consistency for RDF databases. *PVLDB*, 3(1).
- [Nienhuys-Cheng and de Wolf, 1996] Nienhuys-Cheng, S. and de Wolf, R. (1996). Least generalizations and greatest specializations of sets of clauses. *J. Artif. Intell. Res.*
- [Nienhuys-Cheng and de Wolf, 1997] Nienhuys-Cheng, S. and de Wolf, R., editors (1997). *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Computer Science*. Springer.
- [Papailiou et al., 2014] Papailiou, N., Tsoumakos, D., Konstantinou, I., Karras, P., and Koziris, N. (2014). H₂rdf+: an efficient data management system for big RDF graphs. In *SIGMOD*.
- [Perner, 2017] Perner, P. (2017). Mining frequent subgraph pattern over a collection of attributed-graphs and construction of a relation hierarchy for result reporting. In *Advances in Data Mining. Applications and Theoretical Aspects - 17th Industrial Conference, ICDM 2017, New York, NY, USA, July 12-13, 2017, Proceedings*, pages 323–344.
- [Picalausa et al., 2012] Picalausa, F., Luo, Y., Fletcher, G. H., Hidders, J., and Vansummeren, S. (2012). A structural approach to indexing triples. In *ESWC*.
- [Pichler et al., 2010] Pichler, R., Polleres, A., Skritek, S., and Woltran, S. (2010). Redundancy elimination on RDF graphs in the presence of rules, constraints, and queries. In *RR*.
- [Pichler et al., 2013] Pichler, R., Polleres, A., Skritek, S., and Woltran, S. (2013). Complexity of redundancy detection on RDF graphs in the presence of rules, constraints, and queries. *Semantic Web*, 4(4).
- [Plotkin, 1970] Plotkin, G. D. (1970). A note on inductive generalization. *Machine Intelligence*, 5.
- [Plotkin, 1971] Plotkin, G. D. (1971). A further note on inductive generalization. *Machine Intelligence*, 6.
- [Raedt, 2008] Raedt, L. D. (2008). *Logical and relational learning*. Cognitive Technologies. Springer.

- [Ramakrishnan and Gehrke, 2003] Ramakrishnan, R. and Gehrke, J. (2003). *Database management systems*. McGraw-Hill.
- [Robinson, 1965] Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1).
- [Robinson and Voronkov, 2001] Robinson, J. A. and Voronkov, A., editors (2001). *Handbook of Automated Reasoning*. Elsevier and MIT Press.
- [Sherif et al., 2017] Sherif, M. A., Ngomo, A. N., and Lehmann, J. (2017). Wombat - A generalization approach for automatic link discovery. In *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*, pages 103–119.
- [ter Horst, 2005] ter Horst, H. J. (2005). Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. *J. Web Sem.*, 3(2-3):79–115.
- [Urbani et al., 2012] Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., and Bal, H. E. (2012). Webpie: A web-scale parallel inference engine using mapreduce. *J. Web Sem.*, 10.
- [Urbani et al., 2009] Urbani, J., Kotoulas, S., Oren, E., and van Harmelen, F. (2009). Scalable distributed reasoning using mapreduce. In *ISWC*.
- [W3C-RDF, 2014] W3C-RDF (2014). Resource description framework 1.1. <https://www.w3.org/TR/rdf11-concepts>.
- [W3C-RDFS, 2014] W3C-RDFS (2014). RDF 1.1 semantics. <https://www.w3.org/TR/rdf11-mt>.
- [W3C-SPARQL, 2008] W3C-SPARQL (2008). SPARQL protocol and RDF query language. <http://www.w3.org/TR/rdf-sparql-query>.
- [Weiss et al., 2008] Weiss, C., Karras, P., and Bernstein, A. (2008). Hexastore: sextuple indexing for semantic web data management. *PVLDB*, 1(1).
- [Zarri  and Turhan, 2013] Zarri , B. and Turhan, A. (2013). Most specific generalizations w.r.t. general EL-TBoxes. In *IJCAI*.

List of Figures

1.1	Sample RDF graph \mathcal{G}	19
1.2	Sample RDF graph \mathcal{G}'	19
2.1	Sample RDF graphs \mathcal{G}_1 , \mathcal{G}_2 and $\mathcal{G}_{1\text{gg}}$, with $\mathcal{G}_{1\text{gg}}$ the minimal lgg of \mathcal{G}_1 and \mathcal{G}_2 ; their implicit triples (i.e., derived by the rules in Table 1.2) are shown as dashed edges.	28
2.2	Cover graphs of \mathcal{G}_1 and \mathcal{G}_2 in Figure 2.1 (top) and of their saturations w.r.t. the entailment rules in Table 1.2. Triples shown in gray are part of the graph but are redundant w.r.t. those shown in pink.	33
3.1	Sample BGPQs q_1, q_2 and their minimal lgg $q_{1\text{gg}}$	38
3.2	Sample set \mathcal{O} of RDFS ontological constraints	38
3.3	Saturations of the BGPQs q_1 and q_2 from Figure 3.1 w.r.t. \mathcal{O} from Figure 3.2. Triples shown in green are added by saturation.	39
3.4	width=20cm	40
3.5	Cover queries of the BGPQs q_1 and q_2 in Figure 3.1 (top), and of their saturations $q_{1\mathcal{O}}$ and $q_{2\mathcal{O}}$ in Figure 3.3 (bottom). Triples in gray are redundant w.r.t. those in pink.	46
6.1	RDF graph representation of \mathcal{EL} concepts.	73
6.2	$G_{1\text{gg}}$: an lgg of RDF graphs.	74
A.1	DBPEDIA graphs $\setminus \mathcal{O}$ (A).	86
A.2	DBPEDIA graphs $\setminus \mathcal{O}$ (B).	87
B.1	LUBM queries.	89
C.1	DBPEDIA queries.	92
D.1	DBPEDIA extracted ontology \mathcal{O} (1).	94
D.2	DBPEDIA extracted ontology \mathcal{O} (2).	95
D.3	DBPEDIA extracted ontology \mathcal{O} (3).	96
D.4	DBPEDIA extracted ontology \mathcal{O} (4).	97
D.5	DBPEDIA extracted ontology \mathcal{O} (5).	98
D.6	DBPEDIA extracted ontology \mathcal{O} (6).	99

D.7 DBPEDIA extracted ontology \mathcal{O} (7).	100
D.8 DBPEDIA extracted ontology \mathcal{O} (8).	101
D.9 DBPEDIA extracted ontology \mathcal{O} (9).	102
D.10 DBPEDIA extracted ontology \mathcal{O} (10).	103
D.11 DBPEDIA extracted ontology \mathcal{O} (11).	104
D.12 DBPEDIA extracted ontology \mathcal{O} (12).	105
D.13 DBPEDIA extracted ontology \mathcal{O} (13).	106

List of Tables

1.1	RDF & RDFS statements.	18
1.2	Sample RDF entailment rules [W3C-RDFS, 2014].	18
5.1	Characteristics of our test graphs (top) and of their saturations (bottom); times are in ms.	63
5.2	Characteristics of our test BGPQs (top) and of their saturations w.r.t. LUBM constraints (bottom); times are in ms.	64
5.3	Characteristics of our test BGPQs (top) and of their saturations w.r.t. DBpedia constraints (bottom); times are in ms.	64
5.4	Characteristics of cover graph-based lggs of 2 test graphs, w/ or w/o saturation w.r.t. DBpedia RDFS constraints; times are in ms.	65
5.5	Characteristics of cover graph-based lggs of 3 test graphs, w/ or w/o saturation w.r.t. DBpedia RDFS constraints; times are in ms.	65
5.6	Characteristics of cover graph-based lggs of 4 test graphs, w/ or w/o saturation w.r.t. DBpedia RDFS constraints; times are in ms.	66
5.7	Characteristics of cover query-based lggs of test queries, w or w/o using the LUBM RDFS constraints; times are in ms.	67
5.8	Characteristics of cover query-based lggs of 3 test queries, w or w/o using the LUBM RDFS constraints; times are in ms.	67
5.9	Characteristics of cover query-based lggs of 4 test queries, w or w/o using the LUBM RDFS constraints; times are in ms.	67
5.10	Characteristics of cover query-based lggs of test queries, w/ or w/o using the DBpedia RDFS constraints; times are in ms.	68
5.11	Characteristics of cover query-based lggs of 3 test queries, w/ or w/o using the DBpedia RDFS constraints; times are in ms.	68
5.12	Characteristics of cover query-based lggs of 4 test queries, w/ or w/o using the DBpedia RDFS constraints; times are in ms.	68

List of Algorithms

1	Least general anti-unification: <code>lgau</code>	52
2	Cover graph of two RDF graphs: <code>lgg4g</code>	53
3	Cover graph of two RDF graphs: <code>lgg4g-dms</code>	54
4	Cover graph of two RDF graphs: <code>lgg4g-mr</code>	56
5	Cover query of two BGPQs: <code>lgg4q</code>	56

Résumé

La recherche de points communs entre des descriptions de données ou de connaissances est un problème de raisonnement fondamental en Machine Learning, qui a été formalisé par G. Plotkin dans les années 70s sous la forme du calcul du *plus petit généralisant* de ces descriptions.

L'identification des plus petits généralisants a un large panel d'applications qui vont de l'optimisation de requêtes (e.g., pour matérialiser les points communs entre des requêtes lors de la sélection de vues ou pour factoriser leur exécution dans un contexte d'accès concurrentiel), à la recommandation dans le contexte des réseaux sociaux (e.g. pour créer de liens entre des utilisateurs basées sur leurs points communs selon leur profil ou leurs recherches).

Dans cette thèse nous avons revisité la notion du plus petit généralisant dans le contexte de Resource Description Framework (RDF) et le fragment conjonctif de son langage de requêtes associé SPARQL, alias Basic Graph Pattern (BGP) queries. Contrairement à l'état de l'art, nous ne considérons aucune restriction, ni structurelle ni sémantique, sur les graphes et les requêtes. Nos contributions incluent la définition et le calcul des plus petits généralisants dans ces deux formalismes ce qui revient à trouver le plus grand ensemble de points communs entre des bases de données incomplètes et des requêtes conjonctives en présence de contraintes déductives. Nous proposons également une évaluation expérimentale de nos contributions.

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), through exploration (e.g., to classify/categorize datasets and to identify common social graph patterns between organizations (e.g., criminal ones)).

In this thesis we revisit 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. By contrast to the literature, we do not restrict the structure nor semantics of RDF graphs and BGPQs. 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.