HAL
open science

Inferring Same-as Facts from Linked Data: An Iterative Import-by-Query Approach

Mustafa Al-Bakri, Manuel Atencia, Steffen Lalande, Marie-Christine Rousset

To cite this version:

Mustafa Al-Bakri, Manuel Atencia, Steffen Lalande, Marie-Christine Rousset. Inferring Same-as Facts from Linked Data: An Iterative Import-by-Query Approach. Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015), Jan 2015, Austin, Texas, United States. hal-01113463

HAL Id: hal-01113463

https://inria.hal.science/hal-01113463v1

Submitted on 5 Feb 2015

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.

Inferring Same-as Facts from Linked Data: An Iterative Import-by-Query Approach

Mustafa Al-Bakri,[1,2] **Manuel Atencia,**[1,2,3] **Steffen Lalande,**[4] and **Marie-Christine Rousset**[1,2,5]
[1] Univ. Grenoble Alpes, LIG, 38000, Grenoble, France
[2] CNRS, LIG, 38000, Grenoble, France
[3] Inria, 38330, Montbonnot-Saint-Martin, France
[4] Institut National de l'Audiovisuel, 94366, Bry-sur-Marne, France
[5] Institut Universitaire de France, 75005, Paris, France
Mustafa.Al-Bakri@imag.fr, Manuel.Atencia@inria.fr, slalande@ina.fr, Marie-Christine.Rousset@imag.fr

Abstract

In this paper we model the problem of data linkage in Linked Data as a reasoning problem on possibly decentralized data. We describe a novel import-by-query algorithm that alternates steps of sub-query rewriting and of tailored querying the Linked Data cloud in order to import data as specific as possible for inferring or contradicting given target same-as facts. Experiments conducted on a real-world dataset have demonstrated the feasibility of this approach and its usefulness in practice for data linkage and disambiguation.

1 Introduction

Linked Data promotes exposing, sharing and connecting data and knowledge in the Semantic Web (Bizer, Heath, and Berners-Lee 2009). Data linkage is a crucial task in Linked Data. In particular, it is very important to correctly decide whether two URIs refer to the same real-world entity. Most existing approaches are based on numerical methods that return weighted owl:sameAs links, among which those with higher weights are likely (but not guaranteed) to be correct.

In contrast, like a few other works (Saïs, Pernelle, and Rousset 2009; Hogan et al. 2012), we propose a rule-based approach equipped with full reasoning to infer all *certain* same-as facts that are logically entailed from a given set of domain constraints and facts. Our main contribution is a novel algorithm, called Import-by-Query, that enables the scalable deployment of such an approach in the decentralized setting of Linked Data. The main challenge is to identify the data, possibly distributed over several datasets, useful for inferring owl:sameAs and owl:differentFrom facts of interest. Compared to the approach reported in (Hogan et al. 2012), relying on a global import obtained by a breadth-first crawl of the Linked Data cloud, we perform a selective import while guaranteeing completeness for the inference of the targeted owl:sameAs and owl:differentFrom facts.

For doing so, the import-by-query algorithm that we have designed alternates steps of sub-query rewriting and of tailored querying of the Linked Data cloud to import data as specific as possible to infer owl:sameAs and owl:differentFrom facts. It is an extension of the well-known query-subquery algorithm for answering Datalog queries over deductive databases. Experiments conducted on a real-world dataset have demonstrated the feasibility of this approach and its usefulness in practice for data linkage and disambiguation.

Section 2 illustrates our approach by example, while Section 3 positions it w.r.t. existing work. In Section 4 the problem that we consider is formally stated. Our import-by-query algorithm is described in Section 5. Then Section 6 reports on experimental results, and Section 7 concludes the paper.

2 Illustrative Scenario

We describe here a simplified scenario inspired by the task of disambiguation of named entities in a large real-world RDF documentary catalog produced by the French National Audiovisual Institute (INA), and that we have used in our experiments (Section 6).

Figure 1 shows an extract of the INA vocabulary and a sample of RDF triples from the INA dataset.[1] Any person entity is an instance of the class ina:PhysicalPerson, which has two subclasses: ina:Person and ina:VideoPerson. The class ina:Person is used for representing French personalities while ina:VideoPerson is used for identifying person entities that play a role in a video. INA experts want to disambiguate individuals within ina:Person, and link these individuals to the ones of ina:VideoPerson.

Three homonymous persons are described in Figure 1, all named "Jacques Martin": ina:per1, ina:per2 and ina:per3. It is unknown if these entities represent the same or different persons, but some additional information is given: ina:per1 is known to be the presenter of a program recorded in the video ina:vid1 whose title is "Le Petit Rapporteur", whereas ina:per2 and ina:per3 have dates of birth "1933-06-22" and "1921-09-25", respectively.

Our approach to disambiguating the person entities ina:per1, ina:per2 and ina:per3 consists in exploiting domain knowledge and constraints, as well as general properties of owl:sameAs and owl:differentFrom, all this knowledge being expressed in a uniform way by rules. Table 1 shows rules which, for the purpose of this simplified scenario, we can assume they have been validated by INA experts. R1-R3

Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] We have slightly modified the INA vocabulary (e.g. translating French terms into English terms) for the sake of readability.

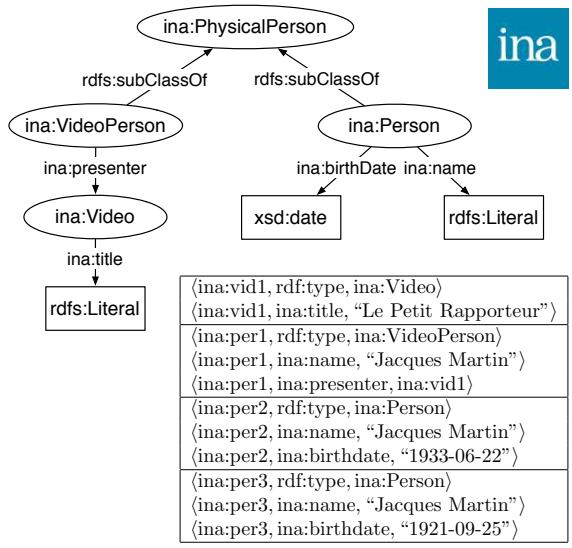

Figure 1: An extract of INA vocabulary and RDF facts.

are domain-specific rules. R1 expresses that ina:birthdate is functional. This rule can be used to infer that ina:per2 and ina:per3 are different because they have different dates of birth. R2 expresses that ina:name and ina:birthdate form a key (within the INA dataset), and R3 the fact that two persons who have the same name and presented programs recorded in videos with the same title must be the same. R2 and R3 indeed could be useful for deciding if ina:per1 refers to the same person as ina:per2 or ina:per3, but some information is missing: the date of birth of ina:per1 is not known, or whether ina:per2 or ina:per3 are presenters and of which programs.

The above missing information can be completed thanks to external data coming from DBpedia. In Figure 2, we show DBpedia facts describing the DBpedia person entity db:per1, and an extract of the DBpedia vocabulary. Rules R4 and R5 in Table 1 translate mappings from the INA and DBpedia vocabularies. Specifically, these mappings state that ina:name and ina:birthdate are equivalent to foaf:name and foaf:birthdate, respectively, and that the composition of ina:presenter and ina:title is equivalent to db:presenter. Let us assume that rules R4 and R5 have been validated by INA experts too. With these rules it can be inferred that db:per1 is the same as ina:per1 because they have the same name and they have presented a program with the same title; and that db:per1 is the same as ina:per2 since they have the same name and birthdate. Therefore, by transitivity of same-as (rule R6 in Table 1), it can be inferred that ina:per1 is the same as ina:per2, and, since ina:per2 is different from ina:per3 then (due to R7) ina:per1 is different from ina:per3 too.

To avoid downloading the complete DBpedia, and, more generally, the whole Linked Open Data (something that is not practical), our import-by-query approach generates, for each targeted owl:sameAs fact, a sequence of external sub-queries as specific as possible to obtain just the missing

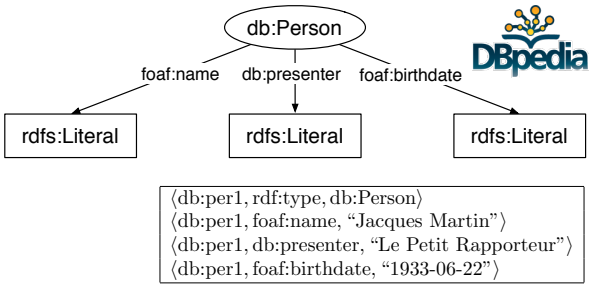

Figure 2: An extract of DBpedia vocabulary and RDF facts.

facts. The external sub-queries generated by our algorithm for the particular query ⟨ina:per1, owl:sameAs, ina:per2⟩ in our example are shown in Figure 3.

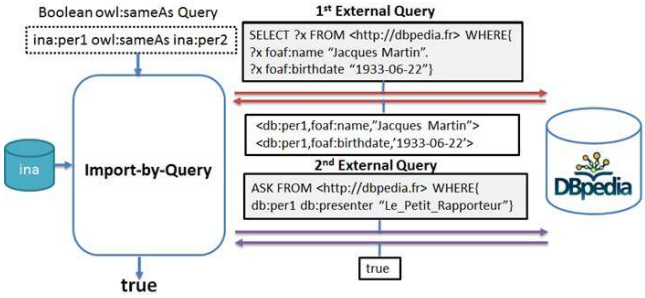

Figure 3: The resultant external sub-queries submitted to DBpedia and their returned answers.

3 Related work

There exists a considerable number of systems that (semi)automatically perform data linkage (Ferrara, Nikolov, and Scharffe 2011). Most of these approaches define or learn metrics to compare entities based on similarities between the values of all or some of their properties. What distinguishes our approach from the majority of these systems is that they are not declarative, with the inherent limitation to be difficult to extend or adapt to new settings with new properties.

There are few existing declarative approaches, in which the comparison methods can be specified by users in the form of (XML) specifications, like in Silk (Volz et al. 2009), or rules, like in LN2R (Saïs, Pernelle, and Rousset 2009) and Hogan et al. (Hogan et al. 2012). Silk specifications can be seen as logical rules along with built-in functions for computing and aggregating similarity degrees between property values. However, these rules are not fully exploited in Silk. More specifically, the possible chaining between rules is not handled by Silk, which makes it incomplete for the task of discovering all the same-as facts that can be logically inferred. As a result, neither Silk nor LIMES (Ngomo and Auer 2011) (similar to Silk in its principles) would discover that ina:p1 is the same as ina:p2 in our illustrative scenario.

LN2R (Saïs, Pernelle, and Rousset 2009) and Hogan et al. (Hogan et al. 2012) come with a complete forward-reasoner and thus guarantee to infer all the same-as facts (and all

R1 : $\langle ?x1$, ina:birthdate, $?b1\rangle$, $\langle ?x2$, ina:birthdate, $?b2\rangle$, $\langle ?b1$, notEqualTo, $?b2\rangle \Rightarrow \langle ?x1$, owl:differentFrom, $?x2\rangle$
R2 : $\langle ?x1$, ina:name, $?n\rangle$, $\langle ?x2$, ina:name, $?n\rangle$, $\langle ?x2$, ina:birthdate, $?b\rangle$, $\langle ?x1$, ina:birthdate, $?b\rangle \Rightarrow \langle ?x1$, owl:sameAs, $?x2\rangle$
R3 : $\langle ?x1$, ina:name, $?n\rangle$, $\langle ?x2$, ina:name, $?n\rangle$, $\langle ?x1$, ina:presenter, $?v1\rangle$, $\langle ?x2$, ina:presenter, $?v2\rangle$, $\langle ?v1$, ina:title, $?t\rangle$, $\langle ?v2$, ina:title, $?t\rangle \Rightarrow \langle ?x1$, owl:sameAs, $?x2\rangle$
R4 : $\langle ?x1$, ina:name, $?n\rangle$, $\langle ?x2$, foaf:name, $?n\rangle$, $\langle ?x1$, ina:presenter, $?v\rangle$, $\langle ?v$, ina:title, $?t\rangle$, $\langle ?x2$, db:presenter, $?t\rangle \Rightarrow \langle ?x1$, owl:sameAs, $?x2\rangle$
R5 : $\langle ?x1$, ina:name, $?n\rangle$, $\langle ?x2$, foaf:name, $?n\rangle$, $\langle ?x1$, ina:birthdate, $?b\rangle$, $\langle ?x2$, foaf:birthdate, $?b\rangle \Rightarrow \langle ?x1$, owl:sameAs, $?x2\rangle$
R6 : $\langle ?x1$, owl:sameAs, $?x2\rangle$, $\langle ?x2$, owl:sameAs, $?x3\rangle \Rightarrow \langle ?x1$, owl:sameAs, $?x3\rangle$
R7 : $\langle ?x1$, owl:sameAs, $?x2\rangle$, $\langle ?x2$, owl:differentFrom, $?x3\rangle \Rightarrow \langle ?x1$, owl:differentFrom, $?x3\rangle$
R8 : $\langle ?x1$, ina:name, $?n1\rangle$, $\langle ?x2$, foaf:name, $?n2\rangle$, $\langle ?n1$, built-in:name-similar, $?n2\rangle$, $\langle ?x1$, ina:birthdate, $?b\rangle$, $\langle ?x2$, foaf:birthdate, $?b\rangle \Rightarrow \langle ?x1$, owl:sameAs, $?x2\rangle$

Table 1: Rules in the INA illustrative scenario.

the different-from facts for LN2R) that can be logically entailed from the rules and facts given as input. Except LN2R, these systems consider using remote data sources to discover links. Unlike our approach, though, the external RDF facts that are loaded to complete local data are obtained either by a global import (of the whole RDF graph of a reference dataset such as DBpedia) or as an incoming data stream produced by a Linked Data crawler. This raises scalability issues that are bypassed either by light-weight incomplete reasoning like in Silk, or by using intensive computational resources requiring clusters of machines like in (Hogan et al. 2012). Instead, our import-by-query algorithm builds iteratively SPARQL queries for importing from external sources in Linked Data the necessary and sufficient data for resolving the link query.

4 Problem statement

We first recall the ingredients of Linked Data and then we define what we call a deductive RDF dataset to capture several ontological constraints expressing data semantics.

RDF datasets in Linked Data

An RDF dataset in Linked Data is defined by a URL u and a set F of RDF facts that are accessible as URL through a query endpoint. We will denote by $ds(u)$ the set F of RDF facts that can be queried at the URL u.

An RDF fact is a triple $t = \langle s, p, o\rangle$ where the subject s is either a URI or a blank node, the predicate p is a URI, and the object o may be either a URI, a blank node or a literal. We will denote the vocabulary used in $ds(u)$ by $voc(u)$, i.e., the names of predicates used to declare triples in the dataset accessible at the URL u.

Queries over RDF datasets in Linked Data

Queries over Linked Data are SPARQL **conjunctive queries** entered through a given query endpoint accessible at a given URL. In this paper, we use a simplified notation for SPARQL queries, and, without loss of generality, we consider that all variables are distinguished.

A query $q(u)$ asked to an RDF dataset identified by (and accessible at) the URL u is a conjunction of **triple patterns** denoted by $TP_1(v_1), \ldots, TP_k(v_k)$ where each triple pattern $TP_i(v_i)$ is a triple $\langle s^v, p^v, o^v\rangle$ in which the subject s^v, the predicate p^v, or the object o^v can be variables: v_i is the set of variables appearing in the triple pattern. Variables are denoted by strings starting by '?'. $TP_i(v_i)$ is a **ground** triple pattern if its set of variables v_i is empty (denoted by $TP_i()$). A ground triple pattern corresponds to a RDF fact. A **boolean query** is a conjunction of ground triple patterns. For instance, the second SPARQL query of Figure 3 in the previous section become in this simplified notation:

q(http://dbpedia.fr):

$\langle$ db:per1, db:presenter, "Le_Petit_Rapporteur" $\rangle$.

The evaluation of a query $q(u) : TP_1(v_1), \ldots, TP_k(v_k)$ over the dataset $ds(u)$ consists in finding substitutions θ assigning the variables in $\bigcup_{i\in[1..k]} v_i$ to constants (i.e., identifiers or literals) such that $TP_1(\theta.v_1), \ldots, TP_k(\theta.v_k)$ are RDF facts in the dataset.

The corresponding answer is equally defined as the tuple of constants assigned by θ to the variables or as the set of corresponding RDF facts $TP_1(\theta.v_1), \ldots, TP_k(\theta.v_k)$ that will be denoted by $\theta.q(u)$. In the remainder of the paper, we will adopt the latter definition. The answer set of the query $q(u)$ against the dataset $ds(u) = F$ is thus defined as:

$$Answer(q(u), F) = \bigcup_{\{\theta|\theta.q(u)\subseteq F\}} \{\theta.q(u)\}$$

For a **boolean query** $q(u)$, either the answer set is not empty and we will say that the query is evaluated to **true**, or it is empty and we will say that it evaluated to **false**.

For a query $q(u)$ to have a chance to get an answer when evaluated over the dataset $ds(u)$, it must be **compatible with** the vocabulary used in this dataset, i.e., (a) the predicates appearing in the triple patterns of $q(u)$ must belong to the set $voc(u)$ of predicates known to occur in $ds(u)$, (b) the URIs appearing as constants in the triple patterns of $q(u)$ must have u as prefix.

In accordance with SPARQL queries allowing different FROM clauses, a conjunctive query can in fact specify several entry points $u_1, \ldots, u_n$ of datasets over which the query to be evaluated. We will denote such a query $q(u_1, \ldots, u_n)$. The above definitions of answers and compatibility can be generalized appropriately by replacing the dataset $ds(u)$ by the union $\bigcup_{i\in[1..n]} ds(u_i)$ of the specified datasets.

Deductive RDF datasets

In order to capture in a uniform way semantic constraints that can be declared on top of a given RDF dataset, but also possibly mappings between local predicates and external predicates within the vocabulary of other datasets, and domain knowledge provided by domain experts, we consider that RDF datasets can be enriched with Datalog rules of the form: $Cond_r \Rightarrow Conc_r$, in which the condition $Cond_r$ is a conjunction of triple patterns (i.e., a conjunctive query) and the conclusion $Conc_r$ is a triple pattern. We consider

safe rules, i.e., rules such that all the variables in the conclusion are also in the condition. Datalog rules on top of RDFS facts capture most of the OWL constraints used in practice, while guaranteeing a polynomial data complexity for reasoning and query answering.

A deductive RDF dataset $dds(u)$ accessible at the URL u is thus a local knowledge base $\langle F, R\rangle$ made of a set of RDF facts F and a set R of rules. The application of rules allows to infer new facts that are logically entailed from $F \cup R$. A rule r can be applied to F if there exists a substitution θ such that $\theta.Cond_r \subseteq F$ and the result of the rule application is $F \cup \{\theta.Conc_r\}$. These new facts can in turn trigger rules and infer additional facts. From a finite set of facts F and a finite set of safe rules R, the set of facts that can be inferred is finite and can be computed as the least fixed point of immediate consequence operator T_R defined as follows:

$$T_R(F) = F \cup \bigcup_{r \in R} \{\theta.Conc_r | \theta.Cond_r \subseteq F\}$$

Let $F_0 = F$, and for every $i \geq 0$, let $F_{i+1} = T_R(F_i)$. There exists a unique least fixed point F_n (denote by $SAT(F, R)$) such that for every $k \geq n$ $F_k = T_R(F_n)$, i.e., there exists a step in the iterative application of the immediate consequence operator for which no new fact is inferred.

The evaluation of a query $q(u) : TP_1(v_1), \ldots, TP_k(v_k)$ over a deductive dataset $dds(u)$ consists in finding substitutions θ such that the facts $TP_1(\theta.v_1), \ldots, TP_k(\theta.v_k)$ can be **inferred** from the deductive dataset, or equivalently belong to the result $SAT(F, R)$ of the facts that can be inferred from F and R:

$$Answer(q(u), \langle F, R\rangle) = Answer(q(u), SAT(F, R))$$

Thus, a **boolean query** $q(u)$ is evaluated to **true** if and only if $q(u) \in SAT(F, R)$, i.e., if and only if $\langle F, R\rangle \vdash q(u)$, where $\vdash$ is the standard notation for logical inference.

Within the vocabulary of a deductive dataset, we distinguish the extensional predicates (EDB predicates for short) that appear in the triplets of the dataset F, from the intentional predicates (IDB predicates) that appear in conclusion of some rules in R. Like in deductive databases, and without loss of generality (i.e., by possibly renaming predicates and adding rules), we suppose that these two sets are disjoint. We will denote ODB predicates the *external* predicates (i.e., defined in a different namespace than the considered deductive dataset) that possibly appear in the dataset or in the rules. These predicates are the core of Linked Data in which a good practice is to re-use existing reference vocabularies. We suppose (again, without loss of generality) that the set of ODB predicates is disjoint from the set of IDB predicates (but not necessarily from the set of EDB predicates).

The problem that we consider can now be formally stated.

Given a deductive dataset $dds(u) = \langle F, R\rangle$, and a boolean query $q(u)$ the local evaluation of which gives an empty answer set (i.e., $\langle F, R\rangle \nvdash q(u)$), we aim to construct a set of external queries $q_1(u_1), \ldots, q_k(u_k)$ for which we can guarantee that the subsets of external facts resulting from their evaluation over the (possibly huge) external datasets are sufficient to answer the initial query. More formally:

$$\langle F \cup_{i \in [1..k]} Answer(q_i(u_i), ds(u_i)), R\rangle \vdash q(u)$$

$$\text{iff } \langle F \cup_{i \in [1..k]} ds(u_i), R\rangle \vdash q(u)$$

The more specific the external queries are, the less external facts have to be added and stored to the local dataset and therefore the more interesting a proposed approach is to solve this problem.

5 The iterative Import-by-Query Algorithm

We now describe the algorithm that we have designed and implemented for solving the problem stated above.
Given an input boolean same-as query q, a deductive dataset $\langle F, R\rangle$, and a set $\bar{u}$ of query entry points to external datasets, Import-by-Query iteratively alternates steps of sub-query rewriting based on backward chaining and of external query evaluation.

Each sub-query rewriting step is realized by an adaptation of the *Query-Subquery* algorithm (Vieille 1986; Abiteboul, Hull, and Vianu 1995) that is a set-oriented memoing backward chaining method (Hinkelmann and Hintze 1993) used in deductive databases for evaluating Datalog programs. This results in the *Query-External-Subquery* (*QESQ* for short) algorithm. For space limitation, here we just explain its main principles, compared to *Query-Subquery*, when applied to a list SG of subgoals. QESQ handles the subgoals built on EDB or IDB predicates exactly like *Query-Subquery*, i.e., iteratively removes subgoals built on EDB predicates if they can be matched with local facts, propagates the corresponding substitutions to the remaining subgoals, replaces a subgoal g built on an IDB predicate by the list of partially instantiated conditions of a rule whose conclusion can be matched to g. As for the subgoals on ODB predicates, they are handled by QESQ before the subgoals on IDB predicates, and if and once all the subgoals built on EDB predicates have been removed, and after the corresponding substitutions are applied to the remaining subgoals in the list. These ODB subgoals are conjuncted to obtain an external query q_{ext}, the compatibility of which must be checked w.r.t. $\bar{u}$ to be considered further. QESQ then treats the remaining list SG_{idb} of subgoals on IDB predicates just as *Query-External-Subquery*, i.e., triggers the recursive call $QESQ(SG_{idb})$. It will return as output either **true** or **false** (if it has enough local information to infer a result to the input boolean query), or a set of external queries that, if compatible with the vocabulary of the given external datasets, are then conjuncted with q_{ext} to constitute the output returned by QESQ(SG). As a result QESQ ($\{q\}$) succeeds in handling locally the goal q using F and R just like *Query-Subquery* and then the process is stopped and the result returned by Import-by-Query is **true** or **false** accordingly, or it produces a set $\{q_1(\bar{u}_1), \ldots, q_k(\bar{u}_k)\}$ of external queries the evaluation of which is likely to bring missing facts to F for proving the goal q using R. If this set is empty, the process is stopped and the result returned by Import-by-Query is **false**.

Each evaluation step simply consists in choosing one of the external query $q_i(\bar{u}_i)$ produced by the sub-query rewriting step and to submit it to Linked Data through the specified

query entry points. The result is either an empty set (negative result) or a set of external facts (positive result) that can be added to the current local dataset. In both cases, the result is memorized in an associated answer table for the sub-query $q_i(\bar{u}_i)$ that will be thus marked as an already processed subgoal for which the (positive or negative) result is known and can be directly exploited later on. If the result is positive, a new iteration of Import-by-Query is started on the same input except for the set of facts F that is enriched with the facts obtained as the result of the evaluation of the external query $q_i(\bar{u}_i)$. If the result is negative, another external query $q_j(\bar{u}_j)$ in the set produced by the current call to QESQ is evaluated. If the evaluation of all the external queries in the set returns 'false', then the process is stopped and the result returned by Import-by-Query on q is **false**.

The termination of the Import-by-Query algorithm relies on the termination of QESQ, which is guaranteed by the same memoing technique as *Query-Subquery* (i.e., by handling goal and answer tables for each ODB and IDB predicate). The soundness and completeness of the Import-by-Query algorithm results from the soundness and completeness of *Query-Subquery* (Vieille 1986) and from the observation that the result produced by *Query-Subquery*, if applied to the same input in which the ODB predicates are just considered as additional EDB predicates, would be the same as the one produced by Import-by-Query. The reason is that the only difference of Import-by-Query is to replace successive matching of atomic goals against the facts by matching *all at once* the atomic goals composing the external queries produced by QESQ. This does not impact the global boolean result of the sequence of goal matching.

Combining forward and backward chaining

Like any backward chaining method, Import-by-Query (and its main component QESQ) re-starts from scratch for each new goal it tries to solve, even if the facts and the rules remain unchanged. The intermediate subgoals generated and handled by QESQ can be simplified if the input rules are replaced by their (partial) instantiations obtained by the propagation of the facts into (the conditions of) the rules.

Fact propagation is a forward chaining method used in inference engines such as RETE (Forgy 1982) for rule-based systems. It avoids redundant evaluation of same conditions appearing in several rules by memorizing, for each fact f, which condition it satisfies in which rule (possibly already partially instantiated by facts previously propagated), and the corresponding variable substitution that is then applied to all the remaining conditions of the rules.

In our setting, we perform fact propagation as a preprocessing step of the Import-by-Query algorithm, by computing at the same time the set $SAT(F, R)$ of facts that can be inferred locally, and the set $PI(F, R)$ of partial instantiations of the rules in R. This forward reasoning step can be summarized as follows, where $SAT(F, R)$ is initialized as F and $PI(F, R)$ is initialized as R:

- **FOR** each f in $SAT(F, R)$
  FOR each rule $Cond_r \Rightarrow Conc_r$ in $PI(F, R)$ having a condition c that can be matched with f, i.e., there exists θ such that $\theta.c = f$
  - ∗ **IF** c is the only condition in $Cond_r$ **THEN** add $\theta.Conc_r$ to $SAT(F, R)$
  - ∗ **ELSE** add to $PI(F, R)$ the rule obtained from $\theta.Cond_r \Rightarrow \theta.Conc_r$ by removing the condition $\theta.c$ (that is satisfied by the fact f).
- Remove from $PI(F, R)$ those rules whose condition contains EDB predicates that are not ODB predicates (and thus cannot be satisfied by local facts).
- **RETURN** $\langle SAT(F, R), PI(F, R)\rangle$

Each partially instantiated rule r_i returned in $PI(F, R)$ is issued from an input rule r in which some conditions have been matched to facts f_1, ..., f_k that have been inferred before (and added to $SAT(F, R)$), and thus allows us to infer the same conclusion as the input rule r on any set of facts including f_1, ..., f_k. The result $SAT(F, R) \cup PI(F, R)$ is then logically equivalent to the input deductive dataset $F \cup R$ for inferring facts on IDB predicates from the union of F and a set OF of external facts (with ODB predicates), i.e. for every fact f an external set of facts OF:

$$\langle F \cup OF, R\rangle \vdash f \text{ iff } \langle SAT(F, R) \cup OF, PI(F, R)\rangle \vdash f$$

Therefore, it can be equivalently used for proving goals by checking whether they belong to $SAT(F, R)$, or for rewriting goals by applying $QESQ$ to the $PI(F, R)$ (instead of the original R).

6 Experiments

We have conducted experiments on a real deductive dataset composed of 35 rules and ∼6 million RDF facts from INA dataset. The rules may be found at http://goo.gl/NfR12w. Most of the 35 rules capture local knowledge in the domain (functional properties and keys declared as schema constraints, and rules provided by INA experts), mappings between INA and DBpedia vocabularies, and general properties of owl:sameAs and owl:differentFrom. Some of the rules of our experiments involve a built-in predicate (called built-in:name-similar) to allow slight differences when comparing literal values corresponding to person names (e.g. R8 in Table 1). This predicate depends on a built-in function which checks if the similarity of the two name strings is above a given threshold. In all our experiments we used edit distance and 0.99 as a threshold. Other built-in predicates involved in the rules are not-equal, less-or-equal, sum, etc.

It is worth noting that the 35 rules can be extended or modified without the need of changing the algorithmic machinery of our approach.

Experimental Goals and Set-Up

The goal of our experiments was threefold: (1) to show that external information available in Linked Open Data is useful to infer owl:sameAs and owl:differentFrom facts within INA referenced persons, and, thus, to disambiguate local homonyms; (2) to assess the gain in reduced imported facts of our Import-by-Query approach compared to approaches based on forward reasoning only; and (3) to evaluate the

runtime of our Import-by-Query algorithm and the possible amortized gain if fact propagation is performed beforehand.

The external datasets from Linked Open Data with which the INA vocabulary shares terms are DBpedia.org and DBpedia.fr. The baseline for evaluating our two first goals is a set of 0.5 million external facts obtained by downloading from DBpedia.org and DBpedia.fr (using their SPARQL endpoints) all the facts about entities having the same name as one of the homonyms in the INA dataset. We applied a preprocessing step on the original INA dataset to keep only the facts on predicates appearing in the rules conditions. The resulting dataset contains almost 1.15 million of RDF facts and will be the INA dataset referred to henceforth.

Our algorithms have been implemented in SWI-Prolog. All the evaluations were done on a machine with an Intel i7 Quad-core processor and 6 GB of memory.

Experimental Results

For evaluating our first goal, we applied (using our forward reasoner) the set of 35 rules to (a) the INA dataset only and (b) the union of the INA dataset with the baseline external facts, and then we compared the number of owl:sameAs and owl:differentFrom facts on INA homonyms we obtained.

The rules applied to the INA dataset only allowed to infer 2 owl:sameAs facts and 108 owl:differentFrom facts, compared to the 4,884 owl:sameAs and 9,764 owl:differentFrom facts inferred when the external facts were added to the process. This clearly demonstrates the benefit of using external information from Linked Open Data for local disambiguation. These resulting 14,648 facts are guaranteed to be correct under the assumption that both rules and data are correct. However, since this is not ensured for DBpedia data, we asked INA experts to evaluate a random sample of 500 of such facts, and all of them were assessed to be true.[2]

The rule expressing owl:sameAs transitivity is crucial for inferring all the owl:sameAs facts that cannot be inferred locally. More generally, full reasoning is very important to discover owl:sameAs and owl:differentFrom facts. In order to show this, we applied Silk to the same two datasets (the INA dataset only, and the union of the INA dataset with the baseline external facts). For doing so, we first had to translate our rules into the Silk specification language. It is not possible, however, to translate into Silk our rules concluding on owl:differentFrom atoms. Thus, we focused on the rules leading to owl:sameAs inference. Among the 4,884 owl:sameAs facts discovered by our full forward reasoner, Silk (which does not perform full reasoning) only discovered 88, i.e. less than 2% of the total. This shows that inference is important for data linkage.

For evaluating our second experimental goal, we took as reference boolean queries the above sample of 500 owl:sameAs and owl:differentFrom facts, and we applied our import-by-query algorithm to each of these boolean queries. The number of external facts imported by our algorithm for all boolean queries was 6,417, which makes, on average, 13 imported facts per boolean query. In contrast, the total number of baseline external facts needed to conclude the boolean queries with the forward reasoner was much higher (∼500,000). This shows that our import-by-query algorithm reduces drastically the number of imported facts needed for disambiguating local data.

Concerning the runtime evaluation, the import-by-query algorithm requires 3 iterations on average — it successively outputs and evaluates 3 external sub-queries (each of them being produced by calling QESQ) — before termination. It takes on average 186 s per boolean query when applied to the initial set of rules and the local dataset. This drops to 7 s when it is applied to the partially instantiated rules obtained by fact propagation beforehand, which means a gain in time of 179 s (∼96%). With respect to the fact propagation, we propagated all facts involving properties of class ina:Person. This took 191 s but it is done only once for all queries, and its cost is amortized very fast, as shown by the above numbers.

7 Conclusion

We have proposed a novel approach for data linkage based on reasoning and adapted to the decentralized nature of the Linked Data cloud. This approach builds on the formal and algorithmic background of answering Datalog queries over deductive databases, that we have extended to handle external rewriting when local answers cannot be obtained. In contrast with existing rule-based approaches for data linkage (Saïs, Pernelle, and Rousset 2009; Hogan et al. 2012) based on forward reasoning to infer same-as facts, Import-by-Query is a backward chaining algorithm that imports *on demand* only external facts useful to infer target same-as facts handled as boolean queries. Our experiments have shown that this approach is feasible and reduces the number of facts needed to be imported. Compared to the depth-first approach sketched in (Abiteboul et al. 2005) for distributed Query-Subquery, our QESQ algorithm generates external rewriting in a breadth-first way.

Performing fact propagation beforehand in order to apply Import-by-Query to a set of more specific rules than the original ones is an optimization close to the ones proposed in QueryPIE (Urbani et al. 2011) for efficient backward reasoning on very large deductive datasets. One important difference, though, is that in the QueryPIE setting, the problem of handling recursive rules can be fully delegated to forward reasoning because all the facts are given and the recursive rules concern a well identified subset of them (so called terminological facts). Another major difference is that Import-by-Query performs query rewriting if no local answer is obtained from the input deductive dataset.

The import-by-query approach in (Grau and Motik 2012) is limited to ABox satisfiability queries used as oracles in Tableau-based reasoning. Compared to the many recent works on ontology-based data access initiated by (Calvanese et al. 2007), in which query rewriting is done independently of the data, we have designed a *hybrid* approach that alternates (external) query rewriting and (local) query answering. We plan to look into this hybrid approach further, in

[2]For copyright reasons, we are not allowed to expose the whole INA dataset. However, 100 of the 500 facts from the sample, and corresponding INA data, may be found at http://goo.gl/amm1fJ and http://goo.gl/zBrqH5.

particular to deal with ontological constraints expressible in Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012).

The interest of our rule-based approach is that it is generic and declarative: new rules can be added without changing the algorithmic machinery. At the moment the rules that we consider are certain. As a result, the same-as facts that they allow to infer are guaranteed to be correct (under the assumption that the input data does not contain erroneous facts). This is crucial to get automatically same-as facts that are certain, in particular when the goal of discovering same-as links is data fusion, i.e. replacement of two URIs by a single one in all relevant facts. Another added-value to get certain same-as and different-from facts is to find noisy data thanks to contradictions. However, in many cases, domain knowledge is not 100% sure such as pseudo-keys (Atencia, David, and Scharffe 2012) and probabilistic mappings (Tournaire et al. 2011). Data itself may be uncertain due to trust and reputation judgements towards data sources (Atencia, Al-Bakri, and Rousset 2013). Handling uncertain domain knowledge should enable to discover more same-as facts that may be true even if inferred with some uncertainty. We plan to extend our rule-based approach to model any kind of data and rules uncertainty as probabilities within the framework of Probabilistic Datalog (Fuhr 2000).

Acknowledgments

This work has been partially supported by the Qualinca project sponsored by the French National Research Agency under grant number ANR-2012-CORD-012, the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01), and by grant TIN2011-28084 of the Ministry of Science and Innovation of Spain, co-funded by the European Regional Development Fund (ERDF).

References

Abiteboul, S.; Abrams, Z.; Haar, S.; and Milo, T. 2005. Diagnosis of asynchronous discrete event systems: datalog to the rescue! In *Proceedings of the Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 13-15, 2005, Baltimore, Maryland, USA*, 358–367. ACM.

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

Atencia, M.; Al-Bakri, M.; and Rousset, M. 2013. Trust in networks of ontologies and alignments. *Journal of Knowledge and Information Systems*. Doi: 10.1007/s10115-013-0708-9.

Atencia, M.; David, J.; and Scharffe, F. 2012. Keys and pseudo-keys detection for web datasets cleansing and interlinking. In *Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings*, volume 7603 of *LNCS*, 144–153. Springer.

Bizer, C.; Heath, T.; and Berners-Lee, T. 2009. Linked Data - The story so far. *International Journal on Semantic Web and Information Systems* 5(3):1–22.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantics* 14:57–83.

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.

Ferrara, A.; Nikolov, A.; and Scharffe, F. 2011. Data linking for the semantic web. *International Journal on Semantic Web and Information Systems* 7(3):46–76.

Forgy, C. 1982. Rete: A fast algorithm for the many patterns/many objects match problem. *Artificial Intelligence* 19(1):17–37.

Fuhr, N. 2000. Probabilistic datalog: implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science* 51(2):95–110.

Grau, B. C., and Motik, B. 2012. Reasoning over ontologies with hidden content: The import-by-query approach. *Journal of Artificial Intelligence Research (JAIR)* 45:197–255.

Hinkelmann, K., and Hintze, H. 1993. Computing cost estimates for proof strategies. In *Extensions of Logic Programming, 4th International Workshop, ELP'93, St. Andrews, U.K., March 29 - April 1, 1993, Proceedings*, volume 798 of *LNCS*, 152–170. Springer.

Hogan, A.; Zimmermann, A.; Umbrich, J.; Polleres, A.; and Decker, S. 2012. Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora. *Journal of Web Semantics* 10:76–110.

Ngomo, A. N., and Auer, S. 2011. LIMES - A time-efficient approach for large-scale link discovery on the web of data. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 2312–2317. IJCAI/AAAI.

Saïs, F.; Pernelle, N.; and Rousset, M. 2009. Combining a logical and a numerical method for data reconciliation. *Journal on Data Semantics* 12:66–94.

Tournaire, R.; Petit, J.; Rousset, M.; and Termier, A. 2011. Discovery of probabilistic mappings between taxonomies: principles and experiments. *Journal on Data Semantics* 15:66–101.

Urbani, J.; van Harmelen, F.; Schlobach, S.; and Bal, H. E. 2011. QueryPIE: backward reasoning for OWL horst over very large knowledge bases. In *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, volume 7031 of *LNCS*, 730–745. Springer.

Vieille, L. 1986. Recursive axioms in deductive databases: the query/subquery approach. In *Expert Database Conf.*, 253–267.

Volz, J.; Bizer, C.; Gaedke, M.; and Kobilarov, G. 2009. Silk - A link discovery framework for the web of data. In *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009.*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org.