Approaches to Relating and Integrating Semantic Data from Heterogeneous Sources

John Keeney
Knowledge & Data Engineering Group & FAME
Computer Science & Statistics, Trinity College
Dublin, Ireland.
John.Keeney@cs.tcd.ie

Aidan Boran, Ivan Bedini, Christopher J. Matheus,
Peter F. Patel-Schneider
Bell Labs, Alcatel-Lucent
Dublin, Ireland* and Murray Hill, USA
{aidan.boran, ivan.bedini, chris.matheus, pfps}
@alcatel-lucent.com

Abstract—Integrating and relating heterogeneous data using inference is one of the cornerstones of semantic technologies and there are a variety of ways in which this may be achieved. Cross source relationships can be automatically translated or inferred using the axioms of RDFS/OWL, via user generated rules, or as the result of SPARQL query result transformations. For a given problem it is not always obvious which approach (or combination of approaches) will be the most effective and few guidelines exist for making this choice. This paper discusses these three approaches and demonstrates them using an “acquaintance” relationship drawn from data residing in common RDF information sources such as FOAF and DBLP datastores. The implementation of each approach is described along with practical considerations for their use. Quantitative and qualitative evaluation results of each approach are presented and the paper concludes with initial suggestions for guiding principles to help in selecting an appropriate approach for integrating heterogeneous semantic data sources.

Keywords- inference; semantic integration; OWL/RDF; rules; query

I. INTRODUCTION

Today, large organizations frequently deploy information and database systems across distinct functional areas of the enterprise (e.g., logistics, sales, production, finance, human resources). The widespread adoption of these systems has created the problem of islands of heterogeneous and distributed information [1][2] complicating the development of integrated processes and applications [1]. Such data integration problems mean that enterprises spend a great deal of time and money attempting to combine information from different sources into a unified format. Frequently cited as the biggest and most expensive challenge that information-technology organizations face, information integration is estimated to consume about 40% of IT budgets [1].

New approaches to integration that formally represent the meaning of data in a system, offer the hope of dealing with semantic heterogeneities. (This is to be contrasted with simple data heterogeneity, where different sources use different identifiers with the same semantic setting.) Indeed, integrating and relating heterogeneous data using inference is one of the cornerstones of semantic technologies. The heterogeneity is often overcome by defining relationships or alignments between ontologies or data models that represent the data sources. There are a variety of ways in which this outcome may be achieved. From a formal semantic perspective, relationships can be 1) automatically inferred using the axioms of RDFS/OWL, 2) identified via user-generated rules or 3) derived from the results of SPARQL queries. These three approaches may not always be semantically equivalent: all that is required here is that an application sees the same results. However, for a given problem it is not always obvious which approach (or combination of approaches) will be most effective and no general guidelines for making this choice exist in the literature.

The work described in this paper is part of a larger research effort to develop a semantic data access methodology and architecture that will enable application programmers to more easily access, query and reason about ontological descriptions of data residing in distributed heterogeneous systems. One aspect of this effort aims at identifying and characterizing general techniques for inferring relationships between data sources; namely the axiomatic approach, the rule-based approach and the query approach. This paper discusses these three approaches and demonstrates them using an “acquaintance” relationship drawn from data residing in common RDF information sources such as those defined using Friend-of-a-Friend (FOAF)\(^1\) and DBLP.\(^2\) The implementation of each approach is described along with some practical considerations for their use. Initial empirical experiments have been conducted to help evaluate each technique and the quantitative and qualitative results are summarized. The paper concludes with discussions on related work, plans for future exploration and some suggestions for guiding principles to help in selecting the most appropriate approach for integrating heterogeneous semantic data sources.

\(^1\) See: http://www.foaf-project.org/
\(^2\) See: http://www.informatik.uni-terier.de/~ley/db/
II. APPROACHES TO RELATING DATA

As discussed in the related work section below, there is a large body of existing work focusing on the discovery, representation, evolution and evaluation of semantic alignment, semantic matching and semantic mapping techniques to support the interoperability of semantic datasets. In this section we discuss three practical application-level approaches for semantic interoperability, motivated by a simple scenario.

A. A Smart Conference Scenario

In the smart conference scenario we imagine a situation where attendees may wish to form de-facto social networks particularly with co-authors and “acquaintances” present at the conference location. A number of semantic models exist to represent relationships such as friendship (e.g., FOAF), authorship (e.g., SWRC), and social networks (e.g., SIOC, COIN); and a number of datasets exist based on these and similar models (e.g., DBLP, foaf-search.net, RKBEExplorer). Even though publication knowledge bases are among the best inter-linked semantic datasets currently available [3], these diverse models and datasets have only limited semantic and data-level interoperability.

While it remains an active research topic, it is currently not possible to efficiently perform automated semantic reasoning operations across distributed semantic datasets, even where those datasets have been aligned. Current common practice is to merge datasets, or parts thereof, into a single semantically aligned knowledge base and perform the necessary inference operations over that datastore, either by materializing the inferred information in the datastore, by performing the necessary operations at query time, or by a combination of the two techniques. In the following sections we describe three approaches for how two large datasets can be aligned and merged to support reasoning, focusing on the aforementioned “acquaintance” relationship between conference attendees and authors.

What matters here, i.e., what an application cares about, is the acquaintance relationship. As far as we are concerned, therefore, the three approaches only need to produce the same acquaintance relations, and is able to do so.

B. Datasets

In this work we examine two semantic models and associated datasets: Friend-of-a-Friend (FOAF) and a semantically enhanced version of the DBLP Computer Science Bibliography (DBLP++) dataset.

The FOAF specification⁵ and core ontology describes people (profiles) on the web, captures links to documents that describe them, and defines linkages between a person and the other people they know. FOAF is most used in two ways: to concisely identify individual people (e.g., in personnel records) or as a mechanism to link people as acquaintances using the “foaf:knows” property. The core FOAF ontology does not itself define any individuals; rather, the ontology is intended to be imported as a schema into individual FOAF documents where it can be used to create a profile or representation of the document owner and to reference FOAF individuals in the FOAF documents of acquaintances. In practice FOAF documents are usually quite small, and of varying quality. Recent initiatives have attempted to gather large FOAF datasets, either as a community project (e.g., FoafSites,⁶ FoafWiki⁷), using web crawlers (e.g., foafsearch.net⁸), or as test datasets for research. For this work the authors combined datasets from a number of sources (BTC 2009 test crawl,⁹ those from the FOAFBulletinBoard on the FoafWiki site, and the authors’ own FOAF documents) resulting in a FOAF dataset with 19,054,802 triples, 340,430 foaf:Person individuals identified by name, and 511,745 occurrences of the foaf:knows relationship.

The DBLP++ dataset, which acts as the backend dataset for the FacetedDBLP search interface, is an enhanced version of the DBLP Computer Science Bibliography.¹⁰ The entire dataset, updated weekly, currently indexes almost 1.5 million publications according to author, venue, journal or series, year, references and subject topic. A SPARQL endpoint to the dataset is supplied by the FacetedDBLP website, which also provides the entire dataset for download in N3 format. The DBLP++ dataset mainly uses the FOAF and SWRC vocabularies to represent the data, with foaf:name used for author names, and foaf:maker used to represent authorship. The resulting DBLP dataset used in this work contains 36,439,753 triples, 850,149 person individuals identified by name, and zero occurrences of the foaf:knows relationship.

For our scenario in which we wish to identify “acquaintances”, there is a key piece of implicit information residing with the DBLP data that we would like to make explicit and integrate with the FOAF data. Specifically, in the case of publications having more than one author (foaf:maker), we can infer that the co-authors are “acquaintances” even though there is no explicit indication that they even know one another. The FOAF specification in fact suggests this type of pattern: “if there exists a foaf:Document that has two people listed as its foaf:makers, then they are probably collaborators of some kind.”

As neither dataset explicitly defines a mechanism to capture a symmetric “acquaintance” relationship between two conference attendees, we introduce an additional ontology where we define a new symmetric OWL object property, “sda:acquaintance”, for our scenario. The challenge then becomes that of mapping the various relationships defined in the FOAF and DBLP datasets onto this sda:acquaintance relationship.

The objective in our Smart Conference scenario is to use this sda:acquaintance relationship to permit conference attendees to track the location and status of other attendees they might be interested in. In the next sections we

---

⁵ See: http://dblp.13s.de/dblp++.php
⁶ See: http://xmlns.com/foaf/0.1/
⁷ See: http://dblp.uni-trier.de/db/dblp.php
⁹ See: http://www.foaf-search.net/
summarize the three approaches we used to semantically derive this relationship.

C. Option 1: Using OWL axioms

In our first approach, all reasoning is performed purely in OWL without the use of user-generated rules or SPARQL queries. This approach provides the advantage of using a single formal representation for knowledge (i.e., OWL/RDF) that requires the use of a single ontological reasoner, of which several are available commercially (e.g., Pellet, BigOWLlim, OntoBroker, BaseVISor) and in open source (e.g., Pellet, SwiftOWLlim, SPIN, Jena). It also tends to be the simplest approach, assuming one has prior experience developing OWL ontologies. Assuming an application is ontologically based in the first place, this approach should generally be the first considered. The major drawback here is that the representational limits of OWL may not permit the formulation of the relationships needed for the intended data integration.

We will now consider how to apply this approach in our Smart Conference scenario to integrate information residing in the FOAF and DBLP++ data sources. Mapping the foaf:knows relationship in the FOAF dataset to the sda:acquaintance relationship using OWL axioms is a simple task of adding a statement that the foaf:knows object property is related to the sda:acquaintance object property using either the owl:equivalentProperty or the rdfs:subPropertyOf axioms, e.g.:

\[(\text{foaf:knows} \sqsubseteq \text{rdfs:subPropertyOf} \text{foaf:maker}) \]

\[(\text{foaf:knows} \sqsubseteq \text{rdfs:subPropertyOf} \text{sda:acquaintance}) \]

or

\[(\text{rdfs:subPropertyOf} \text{foaf:knows}) \]

We chose to represent foaf:knows as a sub-property of sda:acquaintance because rdfs:subPropertyOf construct imposes a looser semantic coupling than does owl:equivalentProperty; i.e., with this approach foaf:knows relationships imply sda:acquaintance relationships (i.e., they are automatically inferred) but nothing is inferred about FOAF constructs from an instance of an sda:acquaintance relationship. When materialized with an OWL inference engine all occurrences of a person individual X being related to a second person Y using foaf:knows will result in a new inferred statement where the first person is related to the second person with the sda:acquaintance relationship, i.e., (X sda:acquaintance Y). A further step of marking the sda:acquaintance property as symmetric will cause the relationship (Y sda:acquaintance X) to be automatically inferred.

Using just OWL axioms to map co-authorship to a “knows” relationship is more involved in the DBLP dataset. This problem can be largely broken into two subparts: transforming the multiple foaf:maker relationships between a document and its authors into a co-maker (co-author) relationship, and deriving the sda:acquaintance relationship from the co-maker/co-authorship relationship. To derive the co-maker/co-authorship relationship between common authors of a given foaf:Document can be achieved by defining an OWL2 property chain for sda:co-maker using foaf:maker and the inverse of foaf:maker; more specifically:

\[\text{SubPropertyOf(} \]
\[\text{ObjectPropertyChain(ObjectInverseOf(} \]
\[\text{foaf:maker} \text{foaf:maker}) \text{ sda:co-maker).} \]

The second step, deriving sda:acquaintance from co-authorship, can be simply achieved by marking the sda:co-maker property chain as a sub-property of the symmetric sda:colleague property, which is itself a sub-property of the symmetric sda:acquaintance property. When these steps are implemented and the ontology is loaded with a small relevant dataset into an OWL reasoner, this approach operates as expected to infer all relevant sda:acquaintance relationships.

D. Option 2: Using user-defined rules

User-defined rules form a common approach to represent mappings or alignments between heterogeneous semantic data sources (e.g., [4]). Here we use Jena rules to implement a rule-based approach to inferring the sda:acquaintance relationship from the foaf:knows and DBLP co-authorship relationships.

For the DBLP dataset we infer the colleague relationship using this rule:

\[\text{[AuthorRule: } \]
\[\text{(?Document foaf:maker ?Person1)} \]
\[\text{(?Document foaf:maker ?Person2)} \]
\[\text{(?Person1 sda:colleague ?Person2)} \]
\[\text{(?Person2 sda:colleague ?Person1)} \]

This rule states that when a document (?Document) has more than one foaf:maker (?Person1 and ?Person2) then infer that the foaf:makers are sda:colleagues (and sda:acquaintances). Depicted graphically:

\[\text{if: } \]
\[\text{?Document} \rightarrow \text{?Person1 then: } \]
\[\text{?Person1 sda:colleague ?Person2} \]

A similar Jena rule can also be defined for FOAF:

\[\text{[FoafRule: } \]
\[\text{(?Person2 foaf:knows ?Person1)} \]
\[\text{(?Person2 sda:acquaintance ?Person1)} \]
\[\text{(?Person2 sda:acquaintance ?Person1)} \]

In order to provide a fair side-by-side comparison of the rule-based approach to the axiomatic approach we intend to test with all native OWL inference disabled, with just a rule engine to evaluate the rules. While this will operate as expected for the DBLP dataset (since all statements are explicitly stated in the dataset) a problem will quickly arise with the FOAF dataset. With FOAF it is common for a single person to be represented by a number of different foaf:Person instances inter-related by the owl:sameAs
property. Without OWL inference, properties for each individual will not be materialized for all other relevant individuals. For this reason we extend the FOAF ruleset with two new rules:

\[
\text{SameAs1: } (\forall x. \text{owl:sameAs} ?y) \land (\forall x. ?p ?o) \rightarrow (\forall y. ?p ?o)
\]

\[
\text{SameAs2: } (\forall x. \text{owl:sameAs} ?y) \land (\forall ?s ?p ?x) \rightarrow (\forall ?s ?p ?y)
\]

Note that these rules replicate two of the \text{owl:sameAs} axioms in OWL 2 RL (eq-rep-s and eq-rep-o).

E. Option 3: Using SPARQL queries

The third approach to integrating the two data sources, is the use of SPARQL to transform semantic data at query time, as presented in [5]. In this approach data is extracted from a datastore as a result of a SPARQL query, and the transformation or alignment is performed using the SPARQL CONSTRUCT statement to build the desired RDF graph based on the results of the query. This combination of query match clauses (i.e., the SPARQL query WHERE statement) and output construction clauses (i.e., the CONSTRUCT statement) can be shown to be equivalent to a rule-based approach [6].

There are at least two factors to consider before adopting this SPARQL approach. Firstly SPARQL is a query language for RDF and not for RDF(S) or OWL data; as a result it cannot be assumed that any underlying RDF(S) or OWL inferences have been materialized prior to the application of a query. This situation means that any necessary materialization of implicit OWL/RDFS triples must be written into the query (similar to how \text{owl:sameAs} axioms were incorporated into the rulebase in the previous section). Secondly, any SPARQL query engine requires an underlying datastore upon which to perform the query, and thereby allowing the SPARQL processing engine to act as a query endpoint.

Fortunately the expressivity of the SPARQL query language is more than sufficient to implement all of the axioms of RDFS and OWL 2 RL (see SPIN for an example of a reasoner built on top of a SPARQL engine) and there is a proliferation of remotely accessible SPARQL query endpoints that can be used in place of a local query engine (which is the approach the authors took for some of the experiments below).

Mapping all of the \text{foaf:knows} relationships for a Person individual in the FOAF dataset to the sda:acquaintance relationship using a pure RDF approach can be achieved using a pure RDF approach. Here the materialization of the symmetric sda:acquaintance relationship is performed in the query. This approach uses a SPARQL WHERE graph pattern to match \text{foaf:knows} triples. Consider the following graph pattern for \text{foaf:Person} individuals with the property \text{foaf:name} is “John Keeney”:

\[
(\forall \text{Person1} \text{foaf:name "John Keeney"}. \\
\text{?friend foaf:knows ?Person1}. \\
\text{?friend foaf:name ?friendname}. )
\]

In the example above the variable “\text{Person1}” will be bound to the \text{foaf:Person} individual with \text{foaf:name} “John Keeney”, variable “\text{?friend}” will be bound to any \text{foaf:Person} individual related to “\text{Person1}” using the \text{foaf:knows} relationship, and variable “\text{?friendname}” will be bound to the “\text{?friend}”’s \text{foaf:name}. Based on the existence of \text{foaf:Person} individuals that match this RDF graph pattern a transformed graph of RDF can be \text{CONSTRUCT}ed and returned as a result:

\[
\text{CONSTRUCT} \{ \\
\text{?Person1 sda:acquaintance ?friend.} \\
\text{?friend sda:acquaintance ?Person1.} \\
\text{?friend foaf:name ?friendname.} \\
\text{?Person1 foaf:name "John Keeney".} \\
\}
\]

(Note: the name string in the graph pattern and CONSTRUCT clauses of the query can be replaced at runtime with any name; this was done programmatically in the experiments. Also, for clarity, this query is a slight simplification of the one actually used as the sda:colleague relationships were also constructed.)

Unfortunately, the example query given thus far has a number of shortcomings. As described earlier, SPARQL is an RDF query language and does not perform any materialization (i.e., inferencing of inferable triples). For this reason the SPARQL query needs to be extended to search for and return not just the \text{foaf:Person} instances (“\text{Person1} and “\text{?friend}”), but all of their \text{owl:sameAs} instances, and all occurrences of \text{foaf:knows} properties for those instances. It is also necessary for the CONSTRUCT clause to fully materialize all \text{sda:acquaintance} relationships, and their inverse, for all returned \text{foaf:Person} instances and their optional \text{owl:sameAs} instances. This makes the FOAF query much more complicated than represented here and even still does not capture the case where an un-materialized chain of transitive \text{owl:sameAs} statements link a number of un-named \text{foaf:Person} individuals for a single person.

For the DBLP dataset the SPARQL queries to map co-authorship to acquaintance require a simpler query as for the FOAF dataset. Here co-authorship can be found with the following graph pattern:

\[
(\forall \text{Person1} \text{foaf:name "Target Name"}. \\
\text{?Doc foaf:maker ?Person1.} \\
\text{?Doc foaf:maker ?Person2.} \\
\text{?Person2 foaf:name ?name2.} )
\]

Similar to the FOAF query above SPARQL variable “\text{Person1}” and “\text{Person2}” represent \text{foaf:Person} instances, while “\text{?Doc}” represents any \text{foaf:Document} they co-authored. Depicted graphically:

```
foaf:maker - - [ ] ?Person1 - - foaf:name - - TargetName

?Document

foaf:maker - - [ ] ?Person2 - - foaf:name - - ?name2
```
Again, as per the FOAF query above the resulting foaf:Person individuals ("?Person1" and "?Person2") of the query can be formulated into RDF results using CONSTRUCT to assert the sda:acquaintance relationship, and its symmetric inverse, as depicted in this pattern graph:

\[
\text{CONSTRUCT} \{ \text{?Person1 foaf:name TargetName} \} \text{ WHERE} \{ \text{?Person1 sda:acquaintence ?Person2} \}.
\]

III. EVALUATION

One objective of this work is to empirically compare and contrast each of the three discussed approaches to relate and integrate semantic data from heterogeneous sources. This section describes some experiments to support this objective with some preliminary results.

A. Experimental setup

In order to evaluate and compare the three interoperability approaches discussed in a qualitative and quantitative manner, N3 and SQL snapshots of the DBLP++ dataset (as of 13/09/2010) were downloaded, a FOAF data set was readied and an evaluation testbed was prepared. Smaller FOAF datasets were created by manually chopping the FOAF data file into progressively smaller subsets. Smaller DBLP datasets were created by re-exporting the SQL snapshot into N3 format while selecting only every \( \frac{1}{n} \)th publication (and associated authors, venues, etc.) according to its unique identifier to produce datasets 1/n in size. Table 1 shows the sizes of these datasets, with “Huge” being the entire data sources. The datasets were then bulk loaded into local JENA TDB triplestores. For comparison purposes the remote FacetedDBLP SPARQL endpoint was also queried.

### Table 1: Dataset sizes (statements)

<table>
<thead>
<tr>
<th></th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
<th>Huge</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLP</td>
<td>5,419,856</td>
<td>10,217,189</td>
<td>19,348,470</td>
<td>36,439,753</td>
</tr>
<tr>
<td>FOAF</td>
<td>1,733,675</td>
<td>5,145,962</td>
<td>13,552,666</td>
<td>19,054,799</td>
</tr>
</tbody>
</table>

While the axiomatic and rule-based approach operates as expected on small scale tests (10-20k statements), when applied to large datasets, such as the entire FOAF and DBLP test datasets, the reasoning process quickly exhausts available memory on even well-provisioned machines. For the FOAF dataset the SPARQL DESCRIBE query is very similar:

\[

When the query above is applied to the DBLP dataset all RDF triples that mention the foaf:Person individual with the foaf:name “John Keeney” (?Author), all foaf:Documents which have ?Author as an author via foaf:maker (?Doc), and all co-authors of ?Author via the ?Doc variable (?Friend). This query can return a substantial number of RDF statements for authors and co-authors with a large number of indexed publications. As a result of the lack of expressivity to filter the results of DESCRIBE queries substantial extra unnecessary information is returned, placing extra overhead on local reasoning engine: e.g., sda:co-maker and sda:acquaintance statements are also created in a pair-wise manner for all co-authors.

However, the example FOAF query given has a number of shortcomings: In any large FOAF datasets a single person may be represented by several foaf:Person instances, some of which may not have the foaf:name property. Even if these foaf:Person instances are interrelated by owl:sameAs assertions SPARQL will not interpret these OWL statements since SPARQL is an RDF query language with no OWL reasoning. For this reason the query needs to be extended to search for and return not just the foaf:Person instances (?Person and ?Friend), but all owl:sameAs instances, and the case where (?Person foaf:knows ?Friend).

SPARQL DESCRIBE queries above do not alter the RDF statements drawn from the individual datastores but rather return relevant subsets. For this reason the RDF statements returned must be loaded into a datastore, materialized and mapped as described (using OWL axioms or rules), with the relevant sda:co-maker and sda:acquaintance properties inferred.

An alternative to such complex DESCRIBE queries would be to use an inference-enabled native datastore (e.g., Parliament, AllegroGraph), whereby OWL axioms such as owl:sameAs are already materialised before or while queries are handled. Indeed, most inference-enabled datastores support the addition of application specific rules whereby semantic mappings/transformations are handled in the datastore itself. However, this work focuses on integrating already existing datasets, where the datastore

---

10 Xen Virt. Machine, 2xQuadCore Xeon E5420 (2.5GHz, 2x6MB, 1333MHz FSB), 12GB RAM, Windows Vista x64, Java J2SE 1.6.0_23-b05x64, Jena 2.6.3, TDB 0.8.7, Pellet 2.2.1, ARQ 2.8.4
11 http://dblp3.xs.de/d2r/sparql
12 http://parliament.semwebcentral.org/
13 http://www.franz.com/agraph/allegrograph/
mechanism cannot be changed. For this reason we specifically chose a datastore (TDB) with no inference capabilities, where all transforms are performed in the application or the query mechanism.

B. Results and Discussion

To evaluate the performance overhead for all three integration approaches discussed above, a series of eight test queries were applied to each of the datasets: i.e. to retrieve for eight people (the five authors of this work and three others\(^{14}\)) all related foaf:Person individuals using the sda:acquaintance property.

The first evaluation metric considered for the eight test queries was a comparison between the total number of statements retrieved from the datastores, and the number of statements resulting from performing the mapping/transformation operations, as shown in Tables 2 and 3. In these two tables bigger numbers generally imply more effort is needed to determine the acquaintance relationship, i.e., to achieve the result needed for the application. This extra information may be of use elsewhere, but it is not germane to immediate needs.

The second evaluation metric for the eight test cases was a comparison between the total number of statements retrieved from the datastores and the time taken to answer the queries (Note: this is not the time taken to fully materialize/transform with Pellet results from the remote DBLP endpoint, with excessive memory usage after 15 minutes).

A number of interesting results can be drawn from the tables above. From Table 2, the number of statements returned for the same SPARQL DESCRIBE query from different DBLP datastores with very similar contents from two different SPARQL engines varies greatly. For the local ‘huge’ TDB/ARQ datastore compared to the very similar remote ‘huge’ D2R datastore, the former returns 10k statements whereas the latter returns 42k statements. Since the amount of filtering is not mandated in the SPARQL specification [7], SPARQL engines implement DESCRIBE query processing differently. This diversity complicates the query authoring process since the query may need to be tuned for different endpoints. This is particularly obvious in Table 2, where the test-bed application failed to fully materialize/transform with Pellet results from the remote DBLP endpoint.

Another result that can be easily derived from the evaluations above is the efficiency and speed of the SPARQL transform approach when compared to the other approaches. However, this comes at the expense of reusability of the query results. As stated, only the template provided in the CONSTRUCT query is used to produce RDF statements to be returned, with all other data ignored. While this may be appropriate for some applications, for more general-purpose data-processing applications a set of different queries or much more complex queries may be required. Further, determining the correct SPARQL query may be difficult or impossible if the relevant (RDFS or, more likely, OWL) ontologies are complex. Despite these drawbacks, for the use-case presented in this paper, the SPARQL transformation approach is clearly superior from a time and resources point of view.

As stated earlier, both reasoning based approaches (OWL axioms and rules) failed utterly to reason with large datastores. Where the results of the DESCRIBE queries above are treated as smaller datasets it becomes clear that such approaches become time- and memory-prohibitive as the number of triples grow above 10k.

A result that cannot be presented in the tables above was the difficulty of parsing and reasoning with the FOAF dataset. Much of the data in the FOAF dataset was manually authored (unlike the DBLP dataset automatically

---

\(^{14}\) Grigoris Antoniou, Marko Grobelnik, Elena Simperl.
output by D2R), and as a result contains proliferations of badly-formed statements and non-compliance with standards and schemas. As a result the FOAF dataset produced thousands of parser and reasoner warnings and errors. Some of these warnings and errors only became apparent at runtime when the data was returned from the datastore via the DESCRIBE query. This seriously impacts the survivability of any application attempting to parse or reason on the DESCRIBE query. This seriously impacts the ’wild’ user-produced RDF data. Aside from typographical errors some of the main issues included: various attempts to handle lists in RDF; the inability to check domains/ranges for some schemas, especially FOAF, where most properties have no defined domains/ranges (e.g., several occurrences of objects specified as foaf:maker of people); and, due to RDF’s open-world assumption / embedded progressive schema definitions, one newly encountered incorrect statement can cause inconsistency in the entire dataset, or trigger a reasoner failure or overload. All of these issues are themselves ongoing research topics, however, the authors found that none of these errors were encountered with the SPARQL transformation approach since all RDF data returned to the application was generated for the SPARQL CONSTRUCT clause, and was guaranteed to be compliant with the template (schema) provided.

One aspect that became apparent during the evaluation of the three integration approaches was the laborious and time-consuming nature of query and rule authoring and debugging. When the order of clauses in the rules and queries was changed (even when the rule/query semantics were not changed), the time overhead of the query or rule could change radically. The timing results given are based on best effort manual tuning of the rules / queries but it is likely that this is sub-optimal. These findings clearly validate ongoing work towards further tool support for rule and query authoring and ongoing research to automatically optimize queries and reasoner rule-bases.

C. Guiding Principles

Each of the approaches presented and evaluated in this paper have their respective advantages and disadvantages. The first approach – directly using (RDFS or) OWL axioms – is the most general purpose and requires the least amount of extra modeling work, but has, as expected high consumption of computational resources, to the point that large datasets cannot be reliably handled. The third approach – using SPARQL queries to directly produce the desired results – is the cheapest computationally, but suffers from being very specific to application needs and also may require considerable effort in constructing correct and effective queries. However, this approach does have the side benefit of being able to better handle mal-formed data. It might appear that the second approach – using rules instead of axioms – would provide most of the benefits of the other two approaches. However, our results indicate that this approach is often no faster than the direct approach, and it does not provide much of the benefits of using queries, either.

Perhaps the most promising approach would be a combination of these approaches, e.g., where the results of a transforming query are further reasoned or materialized in a manner similar to the axiomatic or rule-based approaches. Such an hybrid approach, using a simplified CONSTRUCT query instead of a DESCRIBE, would greatly reduce the amount of data returned from the datastore, clean-up malformed RDF, remove the need to fully materialize the returned data in the query itself, and exploit the reasoning expressivity of a fully functional reasoner.

D. Related Work

In [5], the SPARQL query language is used to process ontology alignments to support the integration of heterogeneous data sources. The authors note three approaches (i.e. ontology axioms, rules and query). It is also noted that the SPARQL query language is not sufficient for a full-fledged mapping language since it does not provide support for aggregates (e.g., max, sum or average of a resource in a constructed graph), individual generation (e.g., to create a URI from object property) and paths (e.g., indefinite composition). Another interesting and promising approach presented in [8] performs SPARQL query rewriting (the pattern-matching WHERE part) to enable data retrieval from heterogeneous data-sources.

Described in [9] is an OWL-DL based mapping system for query answering in an ontology integration system to provide integrated access to a set of information sources. The mappings are specified as correspondences between conjunctive queries over the ontologies. The authors show that all mappings can be expressed in OWL-DL extended with DL-safe rules.

In [10], the authors show that ontology mappings can be used to support the querying of heterogeneous data sources using a common interface and then transferring data from one knowledge base to another. It is assumed that both the ontologies and mappings are lightweight (i.e. many concepts but few relations and axioms). The key disadvantages of the approach are: a) when there are multiple ontology languages it is not clear how to map between ontologies of different languages, b) if represented in the ontology language, the mappings themselves are tightly coupled to the ontology, and c) the expressivity of the ontology language is not adequate for expressing all types of mappings between ontologies. Thus the authors propose a mapping language, an alignment process and tool that is independent from the ontology language.

A number of other mapping languages and representations exist to define correspondences between different ontologies [11]. Besides a number of systems that use proprietary representations, the most popular approach is to use RDF data to represent mappings, in particular the INRIA format [12]. RDF approaches, especially the INRIA format, suffer from the same limitations as the axiom approach in this paper, in particular their inability to represent more complex mappings that cannot be represented in RDF.

Each of the approaches described above can be grouped into those that use ontology axioms, those that use rules and those that use queries. To date, no one mapping approach, format or representation has been generally accepted as a de-facto standard mechanism to integrate semantic data from heterogeneous sources.
IV. CONCLUSIONS AND FUTURE WORK

This paper discusses and evaluates three alternative approaches to integrating data from heterogeneous semantic data sources. While much of the related work deals with the discovery, representation and manipulation of mappings between knowledge bases, this work aims to compare approaches with the view to choosing an appropriate approach by providing both quantitative and qualitative indications of how each of these approaches performs using large datasets; we also provide some initial guidelines on the advantages and disadvantages of each approach.

For this work we have here described the first in a series of experiments. Ongoing work is focusing on the methods to automatically determine which integration approach is most applicable for a given combination of application and datastore characteristics. The objective remains to establish a set of semantic integration best-practice ‘design patterns’ to simplify some of the difficult issues involved with relating and integrating semantic data from the telco management and control planes.

REFERENCES