Conservative Repurposing of RDF Data

Audun Stolpe and Martin G. Skjæveland

Department of Informatics, University of Oslo
{audus,martige}@ifi.uio.no

1 Introduction

The data oriented Web presents us with the challenge of assessing and ensuring the integrity of data across repeated cycles of reuse. Authoritative sources of information on the Web need to take steps to ensure that the primary nature of its information is maintained by consumers. The present paper is concerned with one aspect of this problem, namely the question of when we are licensed to say that data is being transformed, reused or merged in a non-distortive manner. We shall place this problem in the context of RDF and Linked Data and study the problem in relation SPARQL construct queries.

Example 1. The Cultural Heritage Management Office in Oslo maintains a list of architecturally and culturally valuable buildings known as the Yellow List. Expressed in SPARQL, part of the structure of the list is given by the `WHERE` block in the query below; prefixes are assumed set. Note that there is no explicit representation of city or country, and no grouping of related information in the dataset. Suppose we wish to add structure to the dataset by typing subjects as protected buildings, and grouping similar data around typed nodes. This is illustrated by the `CONSTRUCT` block below:

```
CONSTRUCT { ?x a ?type , mat:ProtectedBuilding ;
   vcard:adr [ a vcard:Address ;
               vcard:street-address ?st ; vcard:postal-code ?c ;
               vcard:locality "Oslo" ; vcard:country-name "Norway"@en ] ;
   vcard:geo [ a vcard:Location ;
               vcard:latitude ?lat ; vcard:longitude ?long ] }
  WHERE { ?x a ?type ;
             cul:street ?st ; cul:code ?c ;
             geo:lat ?lat ; geo:long ?long  }
```

This query changes the very structure of the source, however, there is still a principled relationship between the source and target graphs. For instance the property `cul:street` morphs into the sequence of properties `vcard:adr`, `vcard:street-address`. Indeed, the transformation can easily be seen to be *systematic* in the sense that all pairs related in the same manner in the source graph are transformed *uniformly* in terms of the same structural element in the target graph. It is also *non-distortive* in the sense that no other pair of resources are so related. Contrast with the case in which we replace `cul:code` with `vcard:locality` instead of `vcard:postal-code`, whilst keeping everything else as-is. We would then not be able to distinguish between cities and zip-codes in the repurposed graph, and would in that sense have altered the structure of the information. There is therefore warrant for saying that this particular construct query transforms the original graph in a *conservative manner*. In the following we propose a *general* criterion to sort (what may reasonably be considered) conservative from non-conservative ways of repurposing data. Since we take construct queries as the paradigm of RDF repurposing, this means sorting conservative from non-conservative construct queries. We shall do so first for triple to triple transformation and then sketch similar results for more complex chain to chain mappings.

2 Degrees of Conservativeness

We assume the reader is familiar with RDF and SPARQL syntax and semantics, and introduce only a required minimum of notation. Details can be found in [1, 2]. An *RDF graph* G is a set of triples, and L_G denotes its set of resources taken from $\mathcal{U}$, the universe of all resources. We shall refer to predicates as *edges*, and subject and objects indiscriminately as *vertices*. Note that a resource may be both an edge and a vertex in the same graph. As regards queries, we shall consider only the select-project-join fragment of SPARQL. A *select query* is a pair $\langle S, \boldsymbol{x} \rangle$, where S is a *graph pattern* and $\boldsymbol{x}$ a subset of variables in S. We use $\langle S, \boldsymbol{x} \rangle (G)$ to denote the answer to the query over a graph G. A *construct query* is a pair $\langle C, S \rangle$, where C is a *template* and S a graph pattern containing all variables occurring in C. The answer to a construct query $\langle C, S \rangle$ over an RDF graph G is written $\langle C, S \rangle (G)$.

We now turn to the problem of analysing the notion of a conservative construct query. The analysis in this section is limited to the simple case in which a SPARQL construct query transforms RDF triples to RDF triples. Let G be any RDF graph. As a tentative characterisation, we may say that a construct query is conservative if applied to G it evaluates to a graph that conservatively transforms the subgraph of G that matches the pattern in the `SELECT` clause. Obviously, this pushes the question back to what it means for an RDF graph to be a conservative transformation of another. We shall take the existence of a *p-map* from one to the other to provide an adequate criterion:

Definition 1. *Let G and H be RDF graphs. An* RDF homomorphism *from G to H is a function from L_G to L_H such that if $\langle a, p, b \rangle \in G$, then $\langle h(a), h(p), h(b) \rangle \in H$. A function h is a* p-map *from G to H if it is an RDF homomorphism from G to H and $h(a) = a$ for all vertices a of G.*

As homomorphisms, p-maps reflect the structure of the source in the target. A simple consequence of this is that queries over the source can be translated to queries over the target without loss of tuples in the result set. Thus, the existence of a p-map between the source and the target graph may be taken to account for *systematicity* of a construct query. It does not account for *non-distortiveness* however, for which we also need to reflect the structure of the result back into the source. We shall consider three ways of doing that, represented by the following *bounds* on p-maps $h : G \longrightarrow H$:

$$\textbf{Strong:} \quad \langle a, h(p), b\rangle \in H \Rightarrow \langle a, p, b\rangle \in G \tag{p1}$$

$$\textbf{Liberal:} \quad \langle a, h(p), h(b)\rangle \in H \text{ or } \langle h(a), h(p), b\rangle \in H \Rightarrow \langle a, p, b\rangle \in G \tag{p2}$$

$$\textbf{Weak:} \quad \langle h(a), h(p), h(b)\rangle \in H \Rightarrow \langle a, p, b\rangle \in G \tag{p3}$$

Theorem 1. *If $\langle S, \boldsymbol{x}\rangle(G) \neq \emptyset$, S contains no variables as edges and h is a p-map $h : G \longrightarrow H$ bounded by* (p1)*, then $\langle S, \boldsymbol{x}\rangle(G) = \langle h(S), \boldsymbol{x}\rangle(H)$*

Theorem 1 shows that strong boundedness induces a transformation between RDF graphs that is exact in the sense that the diagram in Fig. 1 commutes:

Theorem 2. *Let h be any function on resources. If for all patterns S we have $\langle S, \boldsymbol{x}\rangle(G) = \langle h(S), \boldsymbol{x}\rangle(H)$, then h is a strongly bounded p-map from G to H.*

The class of strongly bounded p-maps thus completely characterises the pairs of graphs for which there is an exact *triple-to-triple* translation of select queries from one to the other. Note that exactness here does not mean that the source and target are isomorphic. The target may contain more information in the form of triples, as long as these triples do not have source edges that map to them. Characterisation results similar to Theorems 1 and 2 may be had for liberal and weak boundedness as well:

$$\begin{tikzcd}
G \arrow[r, "Q"] \arrow[dr, "h"'] & 2^{\mathcal{U}^n} \\
 & H \arrow[u, "h(Q)"']
\end{tikzcd}$$

Figure 1.

Theorem 3. *Let h be any function from $\mathcal{U}$ to itself and suppose $\langle S, \boldsymbol{x}\rangle(G) \neq \emptyset$, where S contains no variables as edges. Then, $\boldsymbol{a} \in \langle h(S), \boldsymbol{x}\rangle(H) \setminus \langle S, \boldsymbol{x}\rangle(G)$ implies $a \notin L_G$ for any $a \in \boldsymbol{a}$ iff h is a p-map from $G \longrightarrow H$ that satisfies* (p2)*. Moreover, $\boldsymbol{a} \in \langle h(S), \boldsymbol{x}\rangle(H) \setminus \langle S, \boldsymbol{x}\rangle(G)$ implies $a \notin L_G$ for* some *$a \in \boldsymbol{a}$ iff h is a p-map from $G \longrightarrow H$ that satisfies* (p3)*.*

By way of intuitive motivation, strongly bounded p-maps do not allow "new" vertices—i.e. vertices that do not occur in the source—in the target to be related in the same way as source vertices in the target are, thus clearly separating old and new. Liberal p-maps *do* allow such information in the target, but only if it is *exclusively* about new vertices. Finally, weak p-maps allow new and old vertices to be related by mapped edges, but old vertices may not be related in new ways. A composition of two bounded p-maps preserves the weakest bound, implying that p-maps counteract *cumulative error in iterated data repurposing.*

SPARQL graph patterns and templates may be considered as RDF graphs in their own right, and the notion of a p-map may be extended accordingly by including variables in the domain and letting the p-map be the identity on those variables. This allows us to prove the following result:

Theorem 4. *Let $\langle C, S\rangle$ be a construct query, where C contains no variables as edges. If h is a p-map from S to C which is bounded by one of* (p1)–(p3)*, then h is a p-map under the same bound from $\langle S, S\rangle(G)$ to $\langle C, S\rangle(G)$.*

Thus, if there is a bounded p-map from the `WHERE` block to the `CONSTRUCT` block in a construct query, then any subgraph that matches the former can be p-mapped with the same bound into the result of the query. By the properties of bounded p-maps, therefore, we are licensed to say that the construct query is a conservative transformation.

3 Generalizing the Conservativeness Criterion

The concept of a p-map may be put to more creative use and expanded to constrain *chain-to-chain* transformations, no longer requiring that pairs of vertices be consistently and non-distortively related by triples—only that they be so related by chains of triples. We call such a transformation a *c-map*. Space allows us only to outline its constructions and results.

$$\begin{tikzcd}
c_G(G^\uparrow) \arrow[r, "h"] & c_H(H^\uparrow) \\
G^\uparrow \arrow[u, "c_G"] \arrow[r, "f"] & H^\uparrow \arrow[u, "c_H"']
\end{tikzcd}$$

Let $G^\uparrow$ denote the *composition* of an RDF graph G, that is, $G^\uparrow$ contains a triple representative for every chain of triples in G such that $G \subseteq G^\uparrow$ and there is a injective *composition function* c_G which takes a chain in G to its representative in $G^\uparrow$. It can be shown that a c-map f from $G^\uparrow$ to $H^\uparrow$ is *constrained*, which is the equivalent of strong boundedness for chains, if and only if a given p-map from $c_G(G^\uparrow)$ to $c_H(H^\uparrow)$, constructed from f, c_G and c_H, is strongly bounded. This ables us repeat the results of Theorem 4 for the generalized conservativeness criterion, and to show that the transformation in Example 1 is indeed conservative.

4 Concluding Remarks

Preliminary research into computational properties reveals that there is a polynomial algorithm for identifying p-maps between two graphs. For c-maps the situation is more complex, since the composition of a graph may be exponentially larger than the graph itself. Yet, this is not a problem for any realistically sized construct query. The theory has been implemented in the Web application Mapper Dan at `http://sws.ifi.uio.no/MapperDan/`. Mapper Dan takes two RDF graphs or a construct query as input, lets the user specify which bounds to apply, and checks whether there is a corresponding map between the two graphs. This map can then be used to translate the source RDF data to the target vocabulary, produce a construct query which reflects the map, or to rewrite other SPARQL queries.

References

1. Marcelo Arenas, Claudio Gutierrez and Jorge Pérez. 'Foundations of RDF Databases'. In: *Reasoning Web*. Ed. by Sergio Tessaris et al. Vol. 5689. Lecture Notes in Computer Science. Springer, 2009, pp. 158–204.
2. Jorge Pérez, Marcelo Arenas and Claudio Gutierrez. *Semantics of SPARQL*. Tech. rep. TR/DCC-2006-17. Universidad de Chile, 2006.