Conservative Repurposing of RDF Data

Audun Stolpe and Martin G. Skjæveland, {audus,martige}@ifi.uio.no, Department of Informatics, University of Oslo

Synopsis

- **Systematic and non-distortive RDF transformation.**
- **Change representation, preserve content.**
- **Counteract cumulative error in iterated data repurposing.**
- **Handles exact, merger, and extending transformations.**
- **Using homomorphisms with back-conditions.**
  - **Considers only RDF graph structure,**
  - **not semantics.**
  - **Only RDF predicates may be transformed,**
  - **RDF subjects and objects must be left untouched.**
- **Applies to SPARQL CONSTRUCT queries.**
- **Prototype web application available.**

Preliminaries

Let $\mathcal{U}$ denote the set of resources, blank nodes and literals.

An *RDF graph* G (or H) is a set of *triples*, written $\langle a, p, b\rangle$, where $a, p, b \in \mathcal{U}$. We use graph terminology and refer to triple subjects and objects collectively as *vertices*, and predicates as *edges*. The set of vertices and the set of edges of a graph need not be disjoint. Let L_G denote the set of vertices and edges occurring in G.

We consider only the select-project-join fragment of SPARQL. The answer to a SPARQL *SELECT query* is a set of tuples of elements from $\mathcal{U}$. A SPARQL *CONSTRUCT query* is written $\langle C, W\rangle$, where C is a `CONSTRUCT`-block and W is a `WHERE`-block. The answer to a SPARQL CONSTRUCT query over an RDF graph G, written $\langle C, W\rangle(G)$, is an RDF graph.

See poster paper (find link in footer) for details and references.

Homomorphisms and bounds

Def. (RDF homomorphism). *An RDF homomorphism h from G to H is a function from L_G to L_H such that if $\langle a, p, b\rangle \in G$, then $\langle h(a), h(p), h(b)\rangle \in H$.*

Def. (p-map). *A function h is a p-map from G to H if h is an RDF homomorphism from G to H and $h(a) = a$ for all vertices a of G.*

Def. (Bounds). *A p-map $h : G \longrightarrow H$ may be bounded by the following conditions:*

$\langle a, p, b\rangle \in G$

if $\langle a, h(p), b\rangle \in H$*; or* (1)

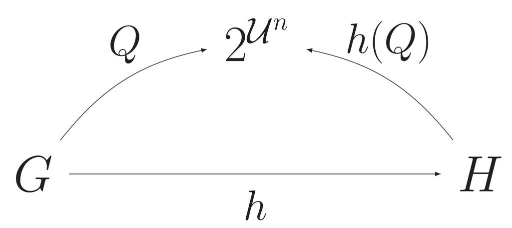
if $\langle a, h(p), h(b)\rangle \in H$ *or* $\langle h(a), h(p), b\rangle \in H$*; or* (2)

if $\langle h(a), h(p), h(b)\rangle \in H$. (3)

We call the bounds (1)–(3) respectively "strong", "liberal" and "weak". Strong ⇒ Liberal ⇒ Weak.

Results

A strongly bounded transformation $h : G \longrightarrow H$ ensures that a SELECT query Q over G returns the *exact same result* as the query $h(Q)$ over H; the diagram commutes:



Similar results exist for liberal and weak bounds.

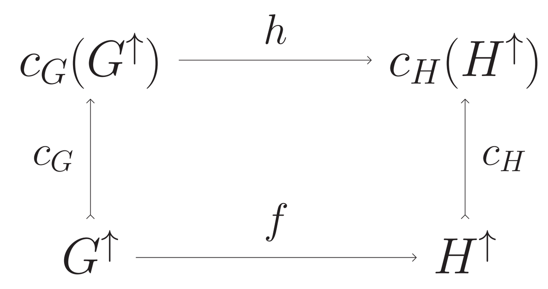
- Strong p-maps do not allow any "new" vertices in the target to be related by "old" edges.
- Liberal p-maps allow also new vertices only to be related by old edges.
- Weak p-maps allow new and old vertices to be related by old edges.
- No p-map allows old vertices to be related by old edges in new ways.

A composition of two bounded p-maps preserves the weakest bound; p-maps *counteract cumulative error in iterated data repurposing*.

Stratified maps: Edges may be transformed under different bounds.

Generalising from triples to chains

c*-maps* generalise triple-to-triple p-maps to *chain-to-chain* transformations. The graph $G^{\uparrow}$ contains a triple-representative for each chain in G, and c_G is the function which takes a chain in G to its triple in $G^{\uparrow}$. A c-map f is strongly chain bound iff h, constructed from f, c_G and c_H, is a strongly bounded p-map.



Example

Let G contain data about culturally valuable buildings in Oslo; an excerpt:

```
ex:Slottet a hva:Bygning;
  hvor:gateadresse "Henrik Ibsens gate 1" ;
  geo:long "10.727928"^^xsd:decimal ;
  geo:lat "59.917667"^^xsd:decimal .
```

Convert G to a fresh H using the vCard vocabulary with the CONSTRUCT query Q:

```
CONSTRUCT { ?x a ?type, cidoc:E25.Man-Made_Feature ;
  vcard:adr [ a vcard:Address ;
             vcard:street-address ?adr ;
             vcard:locality "Oslo" ; ] ;
  vcard:geo [ a vcard:Location ;
    vcard:latitude ?lat ; vcard:longitude ?long ] }
 WHERE { ?x a ?type ; hvor:gateadresse ?adr ;
          geo:lat ?lat ; geo:long ?long }
```

The query represents a conservative repurposing of G.

Why? The main matter of G, address and geographical location data, is mapped with a *strong* bound, e.g., all `geo:lat` relationships from G are mapped to the chain `vcard:geo, vcard:latitude`, ensuring that this data is faithfully preserved in H. "General purpose" vocabulary, as `rdf:type`, is often best mapped under *weak* bounds, allowing one to add types to data in the target H, e.g., `cidoc:E25.Man-Made_Feature` to the buildings from G.

Example contd.

Let G' contain data on the same format as G, but for a different city than Oslo. We want to transform and merge G' with H, now containing the transformed G, using the query Q. Then the address and geographical location data would be mapped under *liberal* bounds, allowing new triples using old edges to be added, but ensuring that the main matters of G and G' are not mixed in H.

SPARQL CONSTRUCT

The `CONSTRUCT`- and `WHERE`-blocks of simple SPARQL CONSTRUCT queries are like RDF graphs. Extending p-maps to the identity function on variables gives us:

Thm. 4. *Let $\langle C, W\rangle$ be a CONSTRUCT query, where W contains no variables as edges. If h is a p-map from W to C bounded by one of (1)–(3), then h is a p-map under the same bound from $\langle W, W\rangle(G)$ to $\langle C, W\rangle(G)$.*

Implementation: Mapper Dan

Implementational features.

- Good: There exists a polynomial algorithm for identifying p-maps between graphs.
- Not so good: When searching for c-maps, constructing the composition of a graph may exponentially increase its size.

Mapper Dan.

- Web app. identifying maps between graphs.
- Reads two RDF graphs or one CONSTRUCT query.
- Find maps according to user's bound requirements.
- Suggests looser bounds if necessary.
- Can apply map to graph, or
- build CONSTRUCT query from map, or
- rewrite queries with map.
- URL: `http://sws.ifi.uio.no/MapperDan/`

UiO : Department of Informatics
University of Oslo

This poster was presented at ISWC 2011. It gives a simplified exposition of that found in the poster paper *Audun Stolpe and Martin G. Skjæveland: Conservative Reporposing of RDF Data* located online at `http://iswc2011.semanticweb.org/fileadmin/iswc/Papers/PostersDemos/iswc11pd_submission_45.pdf`.

Poster by Martin G. Skjæveland.