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, merge, 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.

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 \( (a, p, b) \in G \), then \( (h(a), h(p), h(b)) \in H \).

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 \( a \in G \).

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

\[
\begin{align*}
& (1) \quad \text{if } (a, p, b) \in G \text{ and } h(a), h(b) \in H \Rightarrow h(p) = p \\
& (2) \quad \text{if } (a, p, b) \in G \text{ and } h(a) \in H \text{ or } (h(a), p, b) \in G \Rightarrow h(b) = b \\
& (3) \quad \text{if } (a, p, b) \in G \text{ and } (h(a), p, h(b)) \in H \Rightarrow h(b) = b
\end{align*}
\]

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

Generalising from triples to chains
\( p \)-maps generalise triple-to-triple \( p \)-maps to chain-to-chain transformations. The graph \( G \) 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 \( G' \).

\[
\begin{align*}
& c_G(a) \rightarrow c_G(b) \\
& c_G(p) \rightarrow c_G(H)
\end{align*}
\]

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 \( H \).

Implementation: Mapper Dan

Implementation features.
- Good: There exists a polynomial algorithm for identifying \( p \)-maps between graphs.
- Not so good: When searching for \( p \)-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 loose bounds if necessary.
- Can apply map to graph, or build CONSTRUCT query from map, or rewrite queries with map.

Example
Let \( G \) contain data about culturally valuable buildings in Oslo; an excerpt:

\[
\text{ex:Blottet a hva:Bygning; bvoor:gateadresse "Henrik Ibsens gate 1"; geo:long "10.727928"^^xsd:decimal; geo:lat "59.17667"^^xsd:decimal .}
\]

Convert \( G \) to a fresh \( H \) using the vCard vocabulary with the CONSTRUCT query:

\[
\text{CONSTRUCT \{ ?a \ a vcard:Address ; vcard:street-address ?adr ; vcard:locality "Oslo" ; \} ; vcard:geo [ a vcard:Location ; vcard:latitude ?lat ; vcard:longitude ?long ] \ WHERE \{ ?a \ a vcard:Address ?adr ; geo:lat ?lat ; geo:long ?long \} .}
\]

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 address 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.,

\[
cidoc\E25\manmade\Feature \text{ to the buildings from } G .
\]

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 \( (a, p, b) \), 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.

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

\[
\begin{array}{ccc}
Q & \xrightarrow{h} & h(Q) \\
\downarrow & & \downarrow \\
G & \xrightarrow{h} & H \\
\end{array}
\]

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.

Example
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:

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

Implementation: Mapper Dan

Implementation features.
- Good: There exists a polynomial algorithm for identifying \( p \)-maps between graphs.
- Not so good: When searching for \( p \)-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 loose bounds if necessary.
- Can apply map to graph, or build CONSTRUCT query from map, or rewrite queries with map.

Example
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 \( H \).

Example
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:

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

Implementation: Mapper Dan