Statement Graphs: unifying RDF, RDF-star, and Property Graphs as directed acyclic graphs - Mappings

Ewout Gelling
Eindhoven University of Technology
ewoutgelling1999@gmail.com

George Fletcher
Eindhoven University of Technology
g.h.l.fletcher@tue.nl

Michael Schmidt
Amazon Web Services
schmdtm@amazon.com

1 INTRODUCTION

The mappings in this document make use of the following total functions that relate naturally similar concrete types from the RDF and property graph worlds. They are: $literalvalue : \mathcal{L} \mapsto \mathcal{V}$, $blank2node : \mathcal{B} \mapsto \mathcal{N}$, $IRI2node : \mathcal{I} \mapsto \mathcal{N}$, $IRI2prop : \mathcal{I} \mapsto \mathcal{P}$, and $IRI2label : \mathcal{I} \mapsto 2^{\mathcal{K}}$. There is also an inverse to each of these that achieves the opposite goal, e.g., $value2literal : \mathcal{V} \mapsto \mathcal{L}$, etc. These functions are used to convert the concrete elements associated with leaf nodes of the given Statement Graph. Their entries act as parameters to the mappings. For example, an entry in *literal2value*, relating the literal `"Paris"@en` with property value `"Paris"`. In the below translations, vertex removals are recursive, i.e., when a vertex is removed, all vertices with outgoing edges to that vertex are also removed.

2 FORWARD MAPPINGS

2.1 Property graph to Statement Graph

The following mapping can be applied to property graph $PG = (N, A, P, \delta, \lambda, \sigma, \rho)$ to obtain an LPG-View $G = (V, E, \tau)$

(1) For every $n \in N$:
  (a) Add vertices v_s, v_p, v_o, and v_{st} to V, let $\tau(v_s) = n$, $\tau(v_p) = type$, $\tau(v_o) = \lambda(n)$ and let $\tau(v_{st})$ be an unused element from $\mathcal{S}$
  (b) Add edges $(v_{st}, subject, v_s)$, $(v_{st}, predicate, v_p)$, and $(v_{st}, object, v_o)$ to E.
  (c) For every property $(p_{name}, p_{value}) \in \rho(\sigma(n))$:
    (i) Add vertices $\overline{v_p}$, $\overline{v_o}$ and $\overline{v_{st}}$ to V, let $\tau(\overline{v_p}) = type$, $\tau(\overline{v_o}) = \lambda(n)$ and let $\tau(\overline{v_{st}})$ be an unused element from $\mathcal{S}$
    (ii) Add edges $(\overline{v_{st}}, subject, v_s)$, $(\overline{v_{st}}, predicate, \overline{v_p})$, $(\overline{v_{st}}, object, \overline{v_o})$ to E.
(2) For every $e \in A$, where $\delta(e) = (n_1, n_2)$:
  (a) Add vertices v_s, v_p, v_o, and v_{st} to V, let $\tau(v_s) = n_1$, $\tau(v_p) = \lambda(e)$, $\tau(v_o) = n_2$ and let $\tau(v_{st})$ be an unused element from $\mathcal{S}$
  (b) Add edges $(v_{st}, subject, v_s)$, $(v_{st}, predicate, v_p)$, and $(v_{st}, object, v_o)$ to E.
  (c) For every property $(p_{name}, p_{value}) \in \rho(\sigma(e))$:
    (i) Add vertices $\overline{v_p}$, $\overline{v_o}$ and $\overline{v_{st}}$ to V, let $\tau(\overline{v_p}) = type$, $\tau(\overline{v_o}) = \lambda(n)$ and let $\tau(\overline{v_{st}})$ be an unused element from $\mathcal{S}$
    (ii) Add edges $(\overline{v_{st}}, subject, v_{st})$, $(\overline{v_{st}}, predicate, \overline{v_p})$, $(\overline{v_{st}}, object, \overline{v_o})$ to E.
(3) Merge vertices in V with equal values for τ;

2.2 RDF data set to Statement Graph

The following mapping can be applied to RDF data set $D = \{G_D, (g_i, G_i), \cdots, (g_n, G_n)\}$ to obtain an RDF-View $G = (V, E, \tau)$

(1) Add v_{in} to V, let $\tau(v_{in}) = in$
(2) For all elements $g \in D$:
  (a) Let $(name, graph) = (g_i, G_i)$ if g is a named graph, or let $(name, graph) = (G_{default}, G_D)$ if g is the default graph;
  (b) Add a new vertex v_{graph} to G, let $\tau(v_{graph}) = name$;
  (c) For all RDF triples $(s, p, o) \in graph$:
    (i) Add v_s to G, let $\tau(v_s) = s$;
    (ii) Add v_p to G, let $\tau(v_p) = p$;
    (iii) Add v_o to G, let $\tau(v_o) = o$;
    (iv) Add v_{st} to G, let $\tau(v_{st})$ be an unused element from $\mathcal{S}$
    (v) Add edges $(v_{st}, subject, v_s)$, $(v_{st}, predicate, v_p)$ and $(v_{st}, object, v_o)$ to E;
    (vi) Add v_{mem} to G, let $\tau(v_{mem})$ be an unused element from $\mathcal{S}$
    (vii) Add edges $(v_{mem}, subject, v_{st})$, $(v_{mem}, predicate, v_{in})$ and $(v_{mem}, object, v_{graph})$ to E;
(3) Merge all isomorphic vertices;
(4) Merge vertices in V with equal values for τ.

2.3 RDF-star data set to Statement Graph

The following mapping can be applied to RDF-star data set $D = \{G_D, (g_i, G_i), \cdots, (g_n, G_n)\}$ to obtain an RDF-star-View $G = (V, E, \tau)$

(1) Add v_{in} to V, let $\tau(v_{in}) = in$
(2) For all elements $g \in D$:
  (a) Let $(name, graph) = (g_i, G_i)$ if g is a named graph, or let $(name, graph) = (G_{default}, G_D)$ if g is the default graph;
  (b) Add a new vertex v_{graph} to G, let $\tau(v_{graph}) = name$;
  (c) For all RDF-star triples $(s, p, o) \in graph$:
    (i) If s is an RDF-star triple, then execute steps (i), (ii), (iii), (iv), and (v) on t. Let v_s be the internal vertex obtained in step (v);
    (ii) Else, add v_s to G, let $\tau(v_s) = s$;
    (iii) Add v_p to G, let $\tau(v_p) = p$;
    (iv) If o is an RDF-star triple, then execute steps (i), (ii), (iii), (iv), and (v) on t. Let v_o be the internal vertex obtained in step (v);
    (v) Else, add v_o to G, let $\tau(v_o) = o$;
    (vi) Add v_{st} to G, let $\tau(v_{st})$ be an unused element from $\mathcal{S}$;
    (vii) Add edges $(v_{st}, subject, v_s)$, $(v_{st}, predicate, v_p)$ and $(v_{st}, object, v_o)$ to E;

(viii) Add v_{mem} to G, let $\tau(v_{mem})$ be an unused element from $\mathcal{S}$;
(ix) Add edges $(v_{mem}, subject, v_{st})$, $(v_{mem}, predicate, v_{in})$ and $(v_{mem}, object, v_{graph})$ to E;

(3) Merge all isomorphic vertices;
(4) Merge vertices in V with equal values for τ.

3 BACKWARD MAPPINGS

3.1 Statement Graph to Property graph

The following mapping can be applied to LPG-View $G = (V, E, \tau)$ to obtain a property graph $PG = (N, A, P, \delta, \lambda, \sigma, \rho)$.

(1) Let V_{st} be the set of internal vertices in V. When iterating V_{st}, let v_s, v_p, v_o be vertices with incoming $subject$, $predicate$, $object$ edges from a $v \in V_{st}$
(2) For every $v \in V_{st}$ for which $\tau(v_p) = type$:
   (a) Let node $n = \tau(v_s)$, add n to N, set $\lambda(n) = \tau(v_s)$.
(3) For every $v \in V_{st}$ for which $\tau(v_s) \in \mathcal{N} \wedge \tau(v_o) \in \mathcal{N}$:
   (a) Let node $n_1 = \tau(v_s)$ and $n_2 = \tau(v_o)$, add edge $e = (n_1, n_2)$ to A, set $\lambda(e) = \tau(v_p)$.
(4) For every $v \in V_{st}$ for which $\tau(v_o) \in \mathcal{V}$:
   (a) If $\tau(v_s) \in \mathcal{N}$, let node $n = \tau(v_s)$, add property p to P and entries $\sigma(n) = p$ and $\rho(p) = (\tau(v_p), \tau(v_o))$
   (b) If $\tau(v_s) \in \mathcal{S}$, obtain edge e that was previously created for v_s, add property p to P and entries $\sigma(e) = p$ and $\rho(p) = (\tau(v_p), \tau(v_o))$

3.2 Statement Graph to RDF data set

The following mapping can be applied to RDF-View $G = (V, E, \tau)$ to obtain an RDF data set $D = \{G_D, (g_i, G_i), \cdots, (g_n, G_n)\}$, let D contain only the default graph.

(1) Let V_{st} be the set of internal vertices in V. When iterating V_{st}, let v_s, v_p, v_o be vertices with incoming $subject$, $predicate$, $object$ edges from a $v \in V_{st}$;
(2) For every $v \in V_{st}$ for which $\tau(v_p) \neq in$:
   (a) Create RDF triple $t = (s, p, o)$;
   (b) Let $s = \tau(v_s)$, $p = \tau(v_p)$, $o = \tau(v_o)$;
   (c) Let V_{mem} be the set of vertices with outgoing edges to v;
   (d) For all $v_{mem} \in V_{mem}$, let $\overline{v_o}$ be the vertex with incoming $object$ edge from v_{mem}:
      (i) Let $name = \tau(\overline{v_o})$;
      (ii) If $name = G_D$, add the triple t to the default graph in D.
      (iii) If $name \neq G_D$, add the triple t to the named graph named $name$ in D, if no such graph exists create a new one.

3.3 Statement Graph to RDF-star data set

The following mapping can be applied to RDF-star-View $G = (V, E, \tau)$ to obtain an RDF-star data set $D = \{G_D, (g_i, G_i), \cdots, (g_n, G_n)\}$, let D contain only the default graph.

(1) Let V_{st} be the set of internal vertices in V. When iterating V_{st}, let v_s, v_p, v_o be vertices with incoming $subject$, $predicate$, $object$ edges from a $v \in V_{st}$;
(2) For every $v \in V_{st}$ for which $\tau(v_p) \neq in$:
   (a) Create RDF-star triple $t = (s, p, o)$;
   (b) Let $s = \tau(v_s)$, if $s \in S$ then execute steps (a), (b), (c), and (d) on v_s. Let s be the RDF-star triple obtained in step (d);
   (c) Let $p = \tau(v_p)$;
   (d) Let $o = \tau(v_o)$, if $o \in S$ then execute steps (a), (b), (c), and (d) on v_o. Let o be the RDF-star triple obtained in step (d);
   (e) Let V_{mem} be the set of vertices with outgoing edges to v;
   (f) For all $v_{mem} \in V_{mem}$, let $\overline{v_o}$ be the vertex with incoming $object$ edge from v_{mem}:
      (i) Let $name = \tau(\overline{v_o})$;
      (ii) If $name = G_D$, add the triple t to the default graph in D;
      (iii) If $name \neq G_D$, add the triple t to the named graph named $name$ in D, if no such graph exists create a new one.

4 INTRA-MODEL MAPPINGS

4.1 OneGraph-View to RDF-(star)-View

The following mapping translates the OneGraph-View $G = (V, E, \tau)$ into an RDF-View if the optional step (5) is taken, or an RDF-star-View if the optional step is skipped.

(1) For each leaf node $v \in V$, convert $\tau(v)$ using the appropriate type conversion function if needed, merge nodes with equivalent values for τ;
(2) Add vertices v_{in} and v_{G_D} to V, with $\tau(v_{in}) = in$ and $\tau(v_{G_D}) = G_D$, if such vertices did not already exist in V;
(3) For every internal vertex $v \in V$, add vertex v_{st} to V, $\tau(v_{st}) \in \mathcal{S}$, add edges edges $(v_g, subject, v)$, $(v_g, predicate, in)$, and $(v_g, object, v_{G_D})$ to E;
(4) For every internal vertex $v \in V$, remove vertices in V that are isomorphic to v;
(5) (Optional) For every internal vertex $v \in V$, let v_s, v_p, v_o be vertices with incoming $subject$, $predicate$, $object$ edges from v. If either v_s or v_o is an internal vertex and $\tau(v_p) \neq in$, then remove v;
(6) Remove vertices in V without incoming or outgoing edges.

4.2 OneGraph-View to LPG-View

The following mapping translates the OneGraph-View $G = (V, E, \tau)$ into an LPG-View.

(1) For each leaf node $v \in V$, convert $\tau(v)$ using the appropriate type conversion function if needed, merge nodes with equivalent values for τ. When encountering a $\tau(v) \in \mathcal{I}$:
   (a) if v has an incoming $subject$ or $object$ edge and no incoming $predicate$ edge; Set $\tau(v) = IRI2node(\tau(v))$
   (b) if v has an incoming $predicate$ edge and no incoming $subject$ or $object$ edge; Set $\tau(v) = IRI2label(\tau(v))$
   (c) if v has an incoming $subject$ or $object$ edge and an incoming $predicate$ edge;
      (i) Add node $\overline{v}$ to V, let $\tau(\overline{v}) = IRI2node(\tau(v))$;
      (ii) Let $\tau(v) = IRI2label(\tau(v))$;

(iii) Let all $predicate$ edges point to v, let all $subject$ and $object$ edges point to $\overline{v}$.

(2) For each internal vertex $v \in V$, let v_s, v_p, v_o be vertices with incoming $subject$, $predicate$, $object$ edges from v:
   (a) If v has an incoming edge from some vertex $\overline{v}$, then remove $\overline{v}$ if one of the following conditions hold:
      (i) The incoming edge to v is not labeled $subject$
      (ii) $\tau(v_o) \notin \mathcal{N}$
      (iii) $\overline{v}$ does not have an $object$ edge to a vertex $\overline{v_o}$ with $\tau(\overline{v_o}) \in \mathcal{V}$.

(3) Remove vertices in V without incoming or outgoing edges.