```
# Issue: Two ways of doing reification with CONSTRUCT are not yielding equivalent results

# Create a graph.
CREATE GRAPH ?family, ?grandparents;

# Insert some data into the graph.
INSERT DATA INTO ?family {
  /u<joe> "parent_of"@[] /u<mary> .
  /u<joe> "parent_of"@[] /u<peter> .
  /u<peter> "parent_of"@[] /u<john> .
  /u<peter> "parent_of"@[] /u<eve>
};


# Testing reification in two different ways below:

# 1) Using ";" in CONSTRUCT for reification (partial statements, following docs - "bql.md" file).
# p.s.: it generates 2 new blank nodes (as expected) but no "grandparent"@[] triples.
CONSTRUCT {
  ?ancestor "grandparent"@[] ?grandchildren ; "both_live_in"@[] /city<NY>
}
INTO ?grandparents
FROM ?family
WHERE {
  ?ancestor "parent_of"@[] ?c .
  ?c "parent_of"@[] ?grandchildren
};

# 2) The following version for reification also works (explicit blank node notation).
# p.s.: it is generating only 1 new blank node (in disagreement with the documentation).
# CONSTRUCT {
#   ?ancestor "grandparent"@[] ?grandchildren .
#   _:v "_subject"@[] ?ancestor .
#   _:v "_predicate"@[] "grandparent"@[] .
#   _:v "_object"@[] ?grandchildren .
#   _:v "both_live_in"@[] /city<NY>
# }
# INTO ?grandparents
# FROM ?family
# WHERE {
#   ?ancestor "parent_of"@[] ?c .
#   ?c "parent_of"@[] ?grandchildren
# };

# Verify reification in the ?grandparents graph.
SELECT ?s, ?p, ?o
FROM ?grandparents
WHERE {
  ?s ?p ?o
};

# Drop the graphs.
DROP GRAPH ?family, ?grandparents;
```