Definition of GRAPH*

Olaf Hartig

Linköping University, Linköping, Sweden,
olaf.hartig@liu.se

Abstract. The purpose of this document is to provide a formal definition of the version of the GRAPH operator that we talked about during our meeting, and to also highlight some potential issues with this operator. In this document, I call the operator GRAPH* in order to distinguish it from the standard GRAPH operator as defined for SPARQL at the moment.

1 Preliminaries

For every RDF dataset $D = \{G_{\mathsf{dflt}}, (n_1, G_1), \ldots, (n_m, G_m)\}$ we write $\mathsf{graphs}(D)$ to denote the set of graphs in D. Hence, $\mathsf{graphs}(D) = \{G_{\mathsf{dflt}}, G_1, \ldots, G_m\}$.

Additionally, we write $\mathsf{gnames}(D)$ to denote the set of names of the named graphs in D. Hence, $\mathsf{gnames}(D) = \{n_1, \ldots, n_m\}$. Note that, for any dataset D' that does not contain any named graphs, we have that $\mathsf{gnames}(D') = \emptyset$.

Finally, for every RDF dataset D and every RDF term n that is an IRI or a blank node, we write $\mathsf{graph}(n, D)$ to denote the graph with the name n in D (or the empty graph if D does not contain such a named graph). Formally, $\mathsf{graph}(n, D)$ is defined as follow.

$$\mathsf{graph}(n, D) = \begin{cases} G & \text{if there is a named graph } (n, G) \text{ in } D, \\ \emptyset & \text{else.} \end{cases}$$

2 Auxiliary Functions

This section introduces some auxiliary functions that we will use to define the semantics of the new GRAPH* operator.

As you mentioned during our discussion, you are assuming that an RDF dataset may contain statements about the containment of graphs within other graphs. While the details of these statements need to be specified, for the following formalization we abstract from these details by assuming a function called $\mathsf{contains}$. This function is defined for pairs of the form (D, n) where D is an RDF dataset and n is an IRI or a blank node. The function maps every such pair to a set of IRIs and blank nodes. Hence, for every dataset D and every IRI or blank node n, the function captures the intention that, within a dataset D, the named graph with the name n contains every named graph with a name

$n' \in \mathsf{contains}(D, n)$. Note that $\mathsf{contains}(D, n)$ may be the empty set for some pairs (D, n).

We introduce $\mathsf{contains}^*$ as the transitive closure of $\mathsf{contains}$. That is, for every pair (D, n) with D being an RDF dataset and n an IRI or a blank node, $\mathsf{contains}^*(D, n)$ is a set of IRIs and blank nodes that is defined recursively as follows.

1. $\mathsf{contains}(D, n) \subseteq \mathsf{contains}^*(D, n)$.
2. For every $n' \in \mathsf{contains}^*(D, n)$, it holds that every $\mathsf{contains}(D, n') \subseteq \mathsf{contains}^*(D, n)$.

We introduce a function called $\mathsf{flatten}$ that, for every pair (D, n) with D being an RDF dataset and n an IRI or a blank node, returns the version of the of the named graph with name n in D into which the triples of the contained graphs have been added. Formally, $\mathsf{flatten}(D, n)$ is defined as follows:

$$\mathsf{flatten}(D, n) = \mathsf{graph}(n, D) \cup \left(\bigcup_{n' \in \mathsf{contains}^*(D,n)} \mathsf{graph}(n', D) \right).$$

3 GRAPH*

Now we are ready to define the new operator. We begin with the syntax.

Definition 1. Let P be a graph pattern [1], u be an IRI, and $?x$ be a variable. Then, $(u\ \text{GRAPH}^*\ P)$ and $(?x\ \text{GRAPH}^*\ P)$ are graph patterns.

The semantics of patterns that use the new GRAPH* operator is defined using the following evaluation function.

Definition 2. Let D be an RDF dataset, G be an RDF graph in D (i.e., $G \in \mathit{graphs}(D)$), and P be a graph pattern. The *evaluation* of P over G in the dataset D, denoted by $[\![P]\!]_G^D$, is a set of solution mappings that is defined recursively as follows.

- If P is of the form $(u\ \text{GRAPH}^*\ P')$ where u is an IRI, then

  $$[\![P]\!]_G^D = [\![P']\!]_{G'}^D$$

  where $G' = \mathsf{flatten}(D, u)$.
- If P is of the form $(?x\ \text{GRAPH}^*\ P')$ where $?x$ is a variable, then

  $$[\![P]\!]_G^D = \bigcup_{n \in \mathsf{gnames}(D)} \left([\![P']\!]_{\mathsf{flatten}(D,n)}^D \bowtie \{\mu_{?x \to n}\} \right)$$

  where $\mu_{?x \to n}$ is the solution mapping that maps $?x$ to n and is defined only for $?x$ (i.e., formally, $\mathsf{dom}(\mu_{?x \to n}) = \{?x\}$ and $\mu_{?x \to n}(?x) = n$).
- If P is of any other form, then $[\![P]\!]_G^D$ is as defined by Arenas et al. [1].

4 Examples

Example 1. Consider a dataset $D_{\mathsf{ex1}} = \{G_{\mathsf{dflt}}, (n_1, G_1), (n_2, G_2), (n_3, G_3)\}$ with

$$G_1 = \{(s, p_1, o_1)\},$$
$$G_2 = \{(s, p_2, o_2)\}, \text{ and}$$
$$G_3 = \emptyset.$$

While the exact content of G_{dflt} does not matter for the example, assume the dataset contains statements such that

$$\mathsf{contains}(D_{\mathsf{ex1}}, n_1) = \emptyset,$$
$$\mathsf{contains}(D_{\mathsf{ex1}}, n_2) = \emptyset, \text{ and}$$
$$\mathsf{contains}(D_{\mathsf{ex1}}, n_3) = \{n_1, n_2\}.$$

Moreover, let P_{ex1} be the graph pattern $(n_3\, \mathrm{GRAPH}^* B)$ where $B = \{tp_1, tp_2\}$ is a basic graph pattern (BGP) with the following two triple patterns:

$$tp_1 = (?x, p_1, ?y) \text{ and}$$
$$tp_2 = (?x, p_2, ?z).$$

Then, we have that $[\![P_{\mathsf{ex1}}]\!]^{D_{\mathsf{ex1}}}_{G_{\mathsf{dflt}}} = \{\mu\}$ such that $\mu = \{?x \to s, ?y \to o_1, ?z \to o_2\}$.

Example 2. Consider the same dataset as in the previous example, but now we change the IRI in the graph pattern to a variable. Hence, we consider the graph pattern P_{ex2} of the form $(?g\, \mathrm{GRAPH}^* B)$ where B is the same BGP before. Now we have that $[\![P_{\mathsf{ex2}}]\!]^{D_{\mathsf{ex1}}}_{G_{\mathsf{dflt}}} = \{\mu'\}$ with

$$\mu' = \{?x \to s, ?y \to o_1, ?z \to o_2, ?g \to n_3\}.$$

Example 3. Now consider a dataset $D_{\mathsf{ex2}} = \{G_{\mathsf{dflt}}, (n_1, G_1), (n_2, G_2), (n_3, G_3)\}$ with G_1, G_2, and G_3 as before, but

$$\mathsf{contains}(D_{\mathsf{ex2}}, n_1) = \{n_2\},$$
$$\mathsf{contains}(D_{\mathsf{ex2}}, n_2) = \emptyset, \text{ and}$$
$$\mathsf{contains}(D_{\mathsf{ex2}}, n_3) = \{n_1\}.$$

Using the same graph pattern P_{ex2} of the previous example, we now obtain the following result: $[\![P_{\mathsf{ex2}}]\!]^{D_{\mathsf{ex2}}}_{G_{\mathsf{dflt}}} = \{\mu_1, \mu_2\}$ where

$$\mu_1 = \{?x \to s, ?y \to o_1, ?z \to o_2, ?g \to n_1\} \text{ and}$$
$$\mu_2 = \{?x \to s, ?y \to o_1, ?z \to o_2, ?g \to n_3\}.$$

Note 1. I find it a bit odd to obtain the two solution mappings in the last example, but that is what is supposed to happen according to Definition 2. If that is not the behavior you want, you need to figure out how to change Definition 2 such that it captures your desired behavior.

References

1. M. Arenas, C. Gutierrez, and J. Pérez. On the semantics of SPARQL. In R. D. Virgilio, F. Giunchiglia, and L. Tanca, editors, *Semantic Web Information Management - A Model-Based Perspective*, pages 281–307. Springer, 2009.