Foundations to Query Labeled Property Graphs using SPARQL*

Olaf Hartig

Dept. of Computer and Information Science (IDA), Linköping University, Sweden
olaf.hartig@liu.se

Abstract The RDF*/SPARQL* approach extends RDF and SPARQL with means to capture and to query annotations of RDF triples, which is a feature that is natively available in graph databases modeled as Labeled Property Graphs (LPGs). Hence, the approach presents a step towards making the different graph database models interoperable. This paper takes this step further by providing a solid theoretical foundation for converting LPGs into RDF* data and for querying LPGs using the query language SPARQL*. Regarding the latter, the contributions in this paper consider approaches that materialize the RDF* representation of the LPGs into an RDF*-enabled triplestore as well as approaches in which the queried LPGs may reside in an LPG-specific database system.

1 Introduction

Modern graph database systems can be broadly categorized into two classes: One of these classes comprises systems—called triplestores—that support the Resource Description Framework (RDF) [2] and its query language SPARQL [3]. The foundation of RDF is an abstract data model that represents graph data as triples of the form (subject, predicate, object). A set of such RDF triples is called an RDF graph and it may be illustrated as a directed, edge-labeled multigraph. The systems in the other class support some form of so-called Property Graphs [7], which are directed multigraphs in which nodes and edges may be associated with a set of key-value pairs, called properties. In the most prevalent form of these graphs—called Labeled Property Graphs (LPGs)—the nodes and edges may additionally have labels. Hence, the data that the systems in these different classes manage takes different forms, and there are different query languages for these different forms of graph data. The overall goal of our research is to make these different forms of graph data interoperable.

To this end, in earlier work [5,4] we have introduced an extension of RDF with a notion of key-value-based annotations for triples. These annotations resemble an important feature of LPGs which is not available natively in the RDF data model, namely, the notion of edge properties. Our extension of RDF, which we call RDF*, supports annotations of triples by allowing users to nest RDF triples. That is, in RDF*, nested triples may be used to annotate another triple by directly mentioning this other triple as their subject or their object. Analogous to RDF*, we have extended the RDF query language SPARQL. Our extension, called SPARQL*, enables users to natively express queries for nested RDF* data.

In addition to extending RDF and SPARQL, we envision that RDF* and SPARQL* may also serve as the foundation of a conceptual mediator layer for

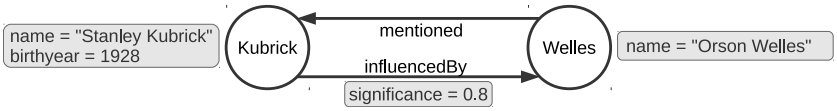

Figure 1. A Labeled Property Graph (LPG) with two nodes and two edges.

integrating RDF data and LPGs, and for using SPARQL* as a common query language. In this context, this paper focuses on the following research question:

How can the query language SPARQL* be used to query LPGs?

We address this question by providing a solid theoretical foundation for converting LPGs into RDF* data and for querying LPGs using SPARQL*. Regarding the latter, the contributions in this paper consider approaches that materialize the RDF* representation of the LPGs in an RDF*-enabled triplestore[1] as well as approaches in which the queried LPGs may reside in an LPG system.

As a basis of this work we introduce a customizable mapping from the LPG model to the RDF* data model. The idea of the mapping is simple: Every edge (including its label) in a given LPG is represented as an ordinary RDF triple in the resulting RDF* data; the same holds for the label of every node, as well as for every node property. Every edge property is represented as a nested triple that contains as its subject the triple representing the corresponding edge.

Example 1.1. An RDF* representation (given in Turtle* format [5]) that is the result of applying the proposed mapping to the LPG in Figure 1 is given as follows:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix p: <http://example.org/property/> .
@prefix r: <http://example.org/relationship/> .

_:b1 rdfs:label "Kubrick" .
_:b1 p:name "Stanley Kubrick" .
_:b1 p:birthyear 1928 .
_:b2 rdfs:label "Welles" .
_:b2 p:name "Orson Welles" .
_:b2 r:mentioned _:b1 .
<<_:b1 r:influencedBy _:b2>> p:significance 0.8 .
```

We emphasize that, while in this example the two nodes of the LPG have been mapped to RDF blank nodes (as represented by the symbols `_:b1` and `_:b2` in the given Turtle* serialization), our proposed mapping also allows users to specify that LPG nodes are mapped to IRIs. Similarly, the mapping gives users the freedom to define patterns for generating IRIs that denote property names and edge labels, respectively. These IRIs become the predicates of triples in the resulting RDF* data (e.g., such as `http://example.org/property/birthyear` in the example). Furthermore, node labels can be defined to be mapped either to literals (as in the example—e.g., `"Kubrick"`) or to IRIs.

[1] Based on RDF*-to-RDF and SPARQL*-to-SPARQL mappings in our earlier work [4], it is even possible to use triplestores that do not yet support RDF* and SPARQL*.

Example 1.2. Given the concepts in this paper, the LPG in Figure 1 may also be queried using SPARQL* queries such as the following (prefix decl. omitted).

```
SELECT ?n ?s WHERE {   ?k p:name "Stanley Kubrick" .
                     <<?k r:influencedBy ?x>> p:significance ?s .
                       ?x p:name ?n .  }
```

Contributions. We make the following concrete contributions in this paper:

1. We introduce a user-configurable direct mapping from LPGs to the RDF* data model, and we provide a formal definition of this mapping. (Section 3)
2. We show that, for edge-unique LPGs (cf. Definition 4.3), our mapping is information preserving, which is the desirable property of being able to convert any resulting RDF* data back into an LPG that is equivalent to the original LPG used as input to the mapping. (Section 4)
3. We define a user-configurable formal semantics for evaluating SPARQL* queries directly over any LPG, including LPGs that are not edge-unique. Hence, based on this semantics, LPGs can be queried using SPARQL* *without* converting them into RDF* data first. (Section 5)
4. We show that our new LPG-specific query semantics coincides with the RDF*-based semantics defined in our earlier work [5]. (also Section 5)

2 Preliminaries

Before focusing on the aforementioned contributions, we need to define the relevant notions of RDF*/SPARQL* and the notion of LPGs that we use in this paper. We begin with the structural part of the RDF* data model. Thereafter, we define the SPARQL* query language by introducing a syntax and a formal query semantics over the RDF* data model. In the end of this section, we define LPGs.

2.1 RDF*

The RDF* data model is an extension of the RDF data model [2]. Consequently, we assume three pairwise disjoint, countably infinite sets: $\mathcal{I}$ (IRIs), $\mathcal{B}$ (blank nodes), and $\mathcal{L}$ (literals). As usual, a tuple $(s, p, o) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$ is an *RDF triple* and a set of RDF triples is called an *RDF graph* [2].

RDF* extends the notion of such triples by permitting a form of nesting; i.e., a triple may have another triple in its first or its third position. Such nesting may be arbitrarily deep. Formally, an *RDF* triple* is defined recursively as follows [5].

Definition 2.1 (RDF* triple). An *RDF* triple* is a 3-tuple such that:

1. Every RDF triple is an RDF* triple.
2. Let t and t' be RDF* triples; for every $s \in (\mathcal{I} \cup \mathcal{B})$, $p \in \mathcal{I}$ and $o \in (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$, the tuples (t, p, o), (s, p, t) and (t, p, t') are RDF* triples.

To denote the (infinite) set of all RDF* triples we write $\mathcal{T}^\star$. Moreover, for any RDF* triple $t = (s, p, o)$ we write $\text{Elmts}^+(t)$ to denote the set of all the RDF terms (IRIs, blank nodes, and literals) and all the RDF* triples that occur within t; i.e., $\text{Elmts}^+(t) = \{s, p, o\} \cup \left\{x \in \text{Elmts}^+(t') \,\middle|\, t' \in \{s, o\} \cap \mathcal{T}^\star\right\}$.

Example 2.1. Consider the two RDF* triples $t_1 = (\mathsf{ex{:}Bob}, \mathsf{ex{:}knows}, \mathsf{ex{:}Alice})$ and $t_2 = (t_1, \mathsf{ex{:}certainty}, 0.8)$. Notice that t_1 is an ordinary RDF triple (i.e., a non-nested RDF* triple) whereas t_2 is a nested RDF* triple. We have that

$$\mathrm{Elmts}^+(t_1) = \{\mathsf{ex{:}Bob}, \mathsf{ex{:}knows}, \mathsf{ex{:}Alice}\}, \text{ and}$$
$$\mathrm{Elmts}^+(t_2) = \{\mathsf{ex{:}Bob}, \mathsf{ex{:}knows}, \mathsf{ex{:}Alice}, t_1, \mathsf{ex{:}certainty}, 0.8\}.$$

Similar to the notion of an RDF graph, we refer to a set of RDF* triples as an *RDF* graph*. For each such graph G we write $\mathrm{T}^+(G)$ to denote the set of all RDF* triples that recursively occur in G, i.e., $\mathrm{T}^+(G) = G \cup \bigcup_{t \in G} \left(\mathrm{Elmts}^+(t) \cap \mathcal{T}^\star\right)$.

Example 2.2. Recall the triples t_1 and t_2 of Example 2.1, and consider an RDF* graph G that only contains t_2; i.e., $G = \{t_2\}$. Then, it holds that $\mathrm{T}^+(G) = \{t_1, t_2\}$.

2.2 SPARQL*

SPARQL* is an RDF*-aware extension of the RDF query language SPARQL. The basic building block of SPARQL queries is a *basic graph pattern* (*BGP*); that is, a finite set of triple patterns, where a *triple pattern* is a tuple of the form $(s, p, o) \in \left(\mathcal{V} \cup \mathcal{I} \cup \mathcal{L}\right) \times \left(\mathcal{V} \cup \mathcal{I}\right) \times \left(\mathcal{V} \cup \mathcal{I} \cup \mathcal{L}\right)$ with $\mathcal{V}$ being a set of query variables that is disjoint from $\mathcal{I}$, $\mathcal{B}$, and $\mathcal{L}$, respectively. On top of BGPs, SPARQL provides many other query operators which are defined based on an algebra [3,6].

SPARQL* extends these concepts by adding the possibility to nest triple patterns. The resulting notion of a *triple* pattern* is defined recursively as follows [5].

Definition 2.2 (triple* pattern). A *triple* pattern* is a 3-tuple such that:

1. Any triple pattern is a triple* pattern.
2. Let tp and tp' be triple* patterns; for every $s \in (\mathcal{V} \cup \mathcal{I} \cup \mathcal{L})$, $p \in (\mathcal{V} \cup \mathcal{I})$ and $o \in (\mathcal{V} \cup \mathcal{I} \cup \mathcal{L})$, tuples (tp, p, o), (s, p, tp) and (tp, p, tp') are triple* patterns.

Similar to the notion of a BGP, we call any finite set of triple* patterns a *BGP**. Note that, by this definition, every ordinary BGP is also a BGP*. While our technical report defines more expressive types of SPARQL* queries [5] (including a complete extension of the W3C specification of SPARQL), in this paper we focus on the BGP* fragment of SPARQL*. However, based on our earlier work [4,5], the results in this paper trivially extend to full SPARQL*.

To define a formal semantics of SPARQL* queries, we recall that the semantics of SPARQL queries is defined based on the notion of solution mappings [3,6]. For SPARQL* we extend this notion to so-called *solution* mappings* that may bind variables not only to IRIs, blank nodes, or literals, but also to RDF* triples. Hence, a *solution* mapping* μ is a partial mapping $\mu : \mathcal{V} \nrightarrow \left(\mathcal{T}^\star \cup \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}\right)$.

Given a solution* mapping μ and a triple* pattern tp, we write $\mu[tp]$ to denote the triple* pattern obtained from tp by replacing the variables in tp according to μ (variables that are not in $\mathrm{dom}(\mu)$ are not replaced). By extension, for any solution* mapping μ and BGP* B, we define $\mu[B] = \bigcup_{tp \in B} \mu[tp]$.

Now we are ready to define an evaluation function that formalizes the semantics of SPARQL* queries over RDF* graphs.

Definition 2.3 (evaluation). Let B be a BGP* and G be an RDF* graph. The *evaluation of B over G*, denoted by $[\![B]\!]_G$, is a set of solution* mappings:

$$[\![B]\!]_G = \{\mu \mid \mathrm{dom}(\mu) = \bigcup_{tp \in B} \mathrm{vars}(tp) \text{ and } \mu[B] \subseteq \mathrm{T}^+(G)\},$$

where $\mathrm{vars}(tp)$ is the set of all variables in a triple* pattern $tp = (s, p, o)$, i.e., $\mathrm{vars}(tp) = (\{s, p, o\} \cap \mathcal{V}) \cup \{x \in \mathrm{vars}(tp') \mid tp' \in \{s, o\} \text{ and } tp' \text{ is a triple}^\star \text{ pattern}\}$.

Example 2.3. Consider the three triple* patterns $tp_1 = (\mathsf{ex{:}Bob}, \mathsf{ex{:}knows}, ?x)$, $tp_2 = (tp_1, \mathsf{ex{:}certainty}, ?c)$, and $tp_3 = (?t, \mathsf{ex{:}certainty}, ?c)$. Then, given the RDF* graph $G = \{t_2\}$ of Example 2.2, we obtain the following query results.

$$[\![\{tp_1\}]\!]_G = \{\mu\} \text{ where } \mu = \{?x \to \mathsf{ex{:}Alice}\},$$
$$[\![\{tp_2\}]\!]_G = \{\mu\} \text{ where } \mu = \{?x \to \mathsf{ex{:}Alice}, ?c \to \mathsf{0.8}\},$$
$$[\![\{tp_3\}]\!]_G = \{\mu\} \text{ where } \mu = \{?t \to t_1, ?c \to \mathsf{0.8}\}.$$

2.3 Labeled Property Graphs

There exist several proposals to define the notion of a Labeled (or unlabeled) Property Graph formally. For our work in this paper we adopt the following definition of Angles et al. [1], which assumes three infinite countable sets: *Labels* (labels), *Props* (property names), and *Values* (property values).

Definition 2.4 (LPG). An *LPG* is a tuple $(V, E, \rho, \lambda, \sigma)$ where:

- V is a finite set of *vertices* (or *nodes*);
- E is a finite set of *edges* such that $V \cap E = \emptyset$;
- $\rho\colon E \to (V \times V)$ is a total function;
- $\lambda\colon (V \cup E) \to \mathit{Labels}$ is a total function;
- $\sigma\colon (V \cup E) \times \mathit{Props} \nrightarrow \mathit{Values}$ is a partial function.

Example 2.4. Figure 1 illustrates the LPG $g = (V, E, \rho, \lambda, \sigma)$ such that

$$V = \{n_{\mathsf{K}}, n_{\mathsf{W}}\}, \qquad E = \{e_{\mathsf{m}}, e_{\mathsf{i}}\}, \qquad \rho = \{e_{\mathsf{m}} \to (n_{\mathsf{W}}, n_{\mathsf{K}}),\ e_{\mathsf{i}} \to (n_{\mathsf{K}}, n_{\mathsf{W}})\},$$
$$\lambda = \{n_{\mathsf{K}} \to \texttt{"Kubrick"},\ n_{\mathsf{W}} \to \texttt{"Welles"},\ e_{\mathsf{m}} \to \texttt{"mentioned"},\ e_{\mathsf{i}} \to \texttt{"influencedBy"}\}, \text{ and}$$
$$\sigma = \{(n_{\mathsf{K}}, \texttt{"name"}) \to \texttt{"Stanley Kubrick"},\ (n_{\mathsf{K}}, \texttt{"birthyear"}) \to \texttt{1928},\ (n_{\mathsf{W}}, \texttt{"name"}) \to \texttt{"Orson Welles"},\ (e_{\mathsf{i}}, \texttt{"significance"}) \to \texttt{0.8}\}.$$

While the given definition of an LPG does not restrict the types of property values, in this paper we assume that all values can be converted to distinct RDF literals. More precisely, we assume a bijective function $\mathsf{vm}\colon \mathit{Values} \to \mathcal{L}_{\mathit{Values}}$ such that $\mathcal{L}_{\mathit{Values}}$ is a set of literals; i.e., $\mathcal{L}_{\mathit{Values}} \subseteq \mathcal{L}$ and $|\mathcal{L}_{\mathit{Values}}| = |\mathit{Values}|$. Note that this assumption rules out LPGs with property values that are of some form of collection (e.g., lists). Hereafter, function vm is called the *property value mapping* and its inverse is denoted by vm^{-1} (i.e., we have that $\mathsf{vm}^{-1}\colon \mathcal{L}_{\mathit{Values}} \to \mathit{Values}$).

3 Property Graph to RDF⋆ Direct Mapping

This section formalizes our proposed mapping from LPGs to RDF⋆ graphs. We refer to such type of a mapping as an *LPG-to-RDF⋆ mapping* which, formally, is a function that maps from the set of all LPGs to the set of all RDF⋆ graphs.

As illustrated in Example 1.1, the idea of our particular LPG-to-RDF⋆ mapping is to represent each node of a given LPG either by a blank node or by an IRI; each edge is represented as an ordinary (non-nested) triple whose subject and object are the blank node or IRI of the adjacent LPG nodes and whose predicate is an IRI that denotes the label of the edge; moreover, node labels and node properties are also represented as ordinary triples, and edge properties are represented as nested triples whose subject is the triple for the corresponding edge.

To formally capture the option for users to customize the mapping by specifying patterns for generating IRIs that denote LPG nodes, labels, and property names we introduce the notion of an LPG-to-RDF⋆ configuration.

Definition 3.1 (LPG-to-RDF⋆ configuration). An *LPG-to-RDF⋆ configuration* is a tuple $(nm, nlm, elm, pm, u_{\mathsf{label}})$ where:

- nm is a function, called *node mapping*, that maps every pair (g, n) consisting of an LPG $g = (V, E, \rho, \lambda, \sigma)$ and a node $n \in V$ to a blank node or an IRI such that for every LPG $g = (V, E, \rho, \lambda, \sigma)$ and every pair of nodes $n \in V$ and $n' \in V$ we have that $nm(g, n) = nm(g, n')$ if and only if $n = n'$,
- nlm is an injective function $nlm\colon \mathit{Labels} \to \mathcal{I} \cup \mathcal{L}$ called *node label mapping*,
- elm is an injective function $elm\colon \mathit{Labels} \to \mathcal{I}$ called *edge label mapping*,
- pm is an injective function $pm\colon \mathit{Props} \to \mathcal{I}$ called *property name mapping*,
- u_{label} is an IRI.

Example 3.1. The property name mapping assumed in Example 1.1 is a function that, for any $pn \in \mathit{Props}$, returns http://example.org/property/*urlenc(pn)* where *urlenc(pn)* is the URL-encoding of the property name pn. Similarly, the edge label mapping returns http://example.org/relationship/*urlenc(ℓ)* for all $\ell \in \mathit{Labels}$. Furthermore, the node mapping in the example returns blank nodes, the node label mapping maps each label to a string literal, and the IRI u_{label} is rdfs:label.

Now we have what we need to define the proposed LPG-to-RDF⋆ mapping.

Definition 3.2 (direct LPG-to-RDF⋆ mapping). Let $c = (nm, nlm, elm, pm, u_{\mathsf{label}})$ be an LPG-to-RDF⋆ configuration. The *c-specific direct LPG-to-RDF⋆ mapping*, denoted by $dm^{c}_{\mathsf{L2R}\star}$, is an LPG-to-RDF⋆ mapping that, for every LPG $g = (V, E, \rho, \lambda, \sigma)$, maps g to an RDF⋆ graph $dm^{c}_{\mathsf{L2R}\star}(g) = G_{\mathsf{nl}} \cup G_{\mathsf{e}} \cup G_{\mathsf{np}} \cup G_{\mathsf{ep}}$ that consists of the following four sets of RDF⋆ triples:

$$G_{\mathsf{nl}} = \big\{\, \mathsf{nl}^{c,g}(n) \mid n \in V \,\big\},$$
$$G_{\mathsf{e}} = \big\{\, \mathsf{e}^{c,g}(e) \mid e \in E \text{ such that } (e, pn) \notin \mathrm{dom}(\sigma) \text{ for all } pn \in \mathit{Props} \,\big\},$$
$$G_{\mathsf{np}} = \big\{\, \mathsf{np}^{c,g}(n, pn) \mid (n, pn) \in \mathrm{dom}(\sigma) \text{ such that } n \in V \,\big\}, \text{ and}$$
$$G_{\mathsf{ep}} = \big\{\, \mathsf{ep}^{c,g}(e, pn) \mid (e, pn) \in \mathrm{dom}(\sigma) \text{ such that } e \in E \,\big\},$$

where

– $\mathsf{nl}^{c,g}$ maps every node $n \in V$ to an RDF* triple

$$\mathsf{nl}^{c,g}(n) = \big(\ nm(g,n),\ u_{\mathsf{label}},\ nlm(\lambda(n))\ \big);$$

– $\mathsf{e}^{c,g}$ maps every edge $e \in E$ with $\rho(e) = (n, n')$ to an RDF* triple

$$\mathsf{e}^{c,g}(e) = \big(\ nm(g,n),\ elm(\lambda(e)),\ nm(g,n')\ \big);$$

– $\mathsf{np}^{c,g}$ maps every $(n, pn) \in \mathrm{dom}(\sigma)$ for which $n \in V$ to an RDF* triple

$$\mathsf{np}^{c,g}(n,pn) = \big(nm(g,n),\ \ pm(pn),\ \mathsf{vm}(\sigma(n,pn))\ \big);$$

– $\mathsf{ep}^{c,g}$ maps every $(e, pn) \in \mathrm{dom}(\sigma)$ for which $e \in E$ to an RDF* triple

$$\mathsf{ep}^{c,g}(e,pn) = \big(\ \mathsf{e}^{c,g}(e),\ pm(pn),\ \mathsf{vm}(\sigma(e,pn))\ \big).$$

Example 3.2. As specified by Definition 3.2, the RDF* graphs resulting from the proposed mapping consist of four subsets, G_{nl}, G_{e}, G_{np}, and G_{ep}. For the RDF* graph in Example 1.1, these subsets are the following (with $b_1 \in \mathcal{B}$ and $b_2 \in \mathcal{B}$):

$$\begin{aligned}
G_{\mathsf{nl}} &= \big\{(b_1, \mathsf{rdfs{:}label}, \text{"Kubrick"}), (b_2, \mathsf{rdfs{:}label}, \text{"Welles"})\big\},\\
G_{\mathsf{e}} &= \big\{(b_2, \mathsf{r{:}mentioned}, b_1)\big\},\\
G_{\mathsf{np}} &= \big\{(b_1, \mathsf{p{:}name}, \text{"Stanley Kubrick"}), (b_1, \mathsf{p{:}birthyear}, 1928),\\
&\qquad (b_2, \mathsf{p{:}name}, \text{"Orson Welles"})\big\},\ \text{and}\\
G_{\mathsf{ep}} &= \big\{((b_1, \mathsf{r{:}influencedBy}, b_2), \mathsf{p{:}significance}, 0.8)\big\}.
\end{aligned}$$

4 Fundamental Properties of the Mapping

In this section we discuss several properties of the proposed mapping.

First, note that our notion of an LPG-to-RDF* configuration distinguishes between node label mappings and edge label mappings (cf. Definition 3.1). While the IRIs returned by an edge label mapping are used as predicates of triples (see function $\mathsf{e}^{c,g}$ in Definition 3.2 and, for instance, set G_{e} in Example 3.2), a node label mapping specifies the objects to use in triples that capture node labels (see function $\mathsf{nl}^{c,g}$ in Definition 3.2 and set G_{nl} in Example 3.2). Note furthermore that, by the definition of node label mappings, such objects in triples for node labels do not need to be literals—as in the examples above—but they may also be IRIs instead. A specific use case for the latter is to carry over type information when mapping LPGs whose node labels represent types and to make the semantics of this typing explicit in the resulting RDF* data.

Example 4.1. Consider the LPG in Figure 2 in which the node labels capture the types of the respective nodes. We may use an LPG-to-RDF* configuration with the same *nm*, *elm*, and *pm* as before (cf. Example 3.1). However, as node label mapping *nlm*, we may use a function that, for every label $\ell \in \mathit{Labels}$, returns

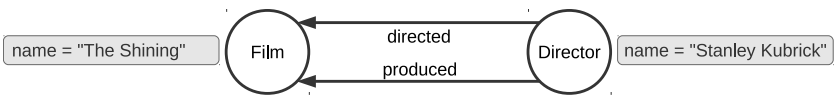

Figure 2. An LPG in which the node labels are used to capture type information.

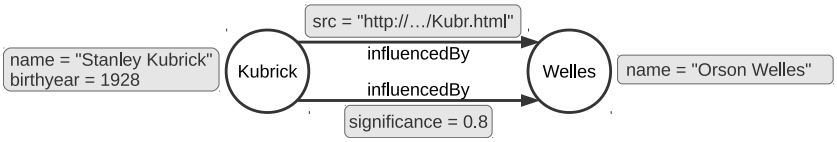

Figure 3. An LPG that is not edge-unique.

the IRI http://example.org/type/*urlenc(*ℓ*)*, and as u_{label} we may use the IRI rdf:type. Then, the result of applying the proposed mapping to the LPG in Figure 2 is the following RDF* graph (with $b_1 \in \mathcal{B}$ and $b_2 \in \mathcal{B}$):

$$G = \{(b_1, \mathsf{rdf{:}type}, \mathsf{http://example.org/type/Film}), (b_1, \mathsf{p{:}name}, \text{"The Shining"}),$$
$$(b_2, \mathsf{rdf{:}type}, \mathsf{http://example.org/type/Director}), (b_2, \mathsf{p{:}name}, \text{"Stanley Kubrick"}),$$
$$(b_2, \mathsf{r{:}directed}, b_1), (b_2, \mathsf{r{:}produced}, b_1)\}.$$

Given this resulting data and an accompanying ontology about relationships between the types and predicates used, it is now even possible to automatically infer additional information using standard RDFS or OWL reasoning. For instance, we may infer that Kubrick was not only a film director but also a film producer.

Another noteworthy (and desirable) property of the proposed mapping is that the mapping is linear. More precisely, based on Definition 3.2 (see also Properties 3a–3d in Note 1 below), it is easy to verify that, for every LPG g, the size of the resulting RDF* graph $dm^c_{\mathsf{L2R}^\star}(g)$ is linear in the size of g.

As a basis for discussing another property, consider the following example.

Example 4.2. Figure 3 illustrates a variation of the initial example LPG (cf. Figure 1) in which we have two edges with the same label, connecting the same nodes, but with different edge properties. The G_{ep} subset of the RDF* graph that the proposed mapping would return in this case contains two nested triples:

$$G_{\mathsf{ep}} = \big\{(t, \mathsf{p{:}src}, \text{"http://.../Kubr.html"}), (t, \mathsf{p{:}significance}, 0.8)\big\},$$

where t denotes the triple $(b_1, \mathsf{r{:}influencedBy}, b_2)$. Based on these two nested triples it is not possible anymore to notice that this data captures two separate edges. The mapping has collapsed the two edges into a single triple, namely, t.

The limitation revealed by the example indicates that there are cases in which applying the proposed mapping results in a loss of some information. This observation brings us to the following question: In which cases does the mapping guarantee to preserve all the information captured in the given LPG?

To answer this question formally, we need to define the *information preservation* property [8] for LPG-to-RDF* mappings. Informally, this property requires the existence of another mapping based on which it is possible to convert any resulting RDF* graph back into an LPG that is *equivalent* to the original LPG. Since, by definition of the LPG model, the identity of nodes and edges is local to any given LPG, we define equivalence of LPGs based on a structure-preserving bijective mapping from nodes to nodes and from edges to edges.

Definition 4.1 (equivalence of LPGs). Two LPGs $g = (V, E, \rho, \lambda, \sigma)$ and $g' = (V', E', \rho', \lambda', \sigma')$ are *equivalent*, denoted by $g \equiv g'$, if there exists a bijection $b : (V \cup E) \to (V' \cup E')$ such that the following six properties holds:

1. for every $n \in V$ it holds that $b(n) \in V'$;
2. for every $e \in E$ it holds that $b(e) \in E'$;
3. for every $e \in E$ it holds that $\rho'(b(e)) = (b(n), b(n'))$ such that $\rho(e) = (n, n')$;
4. for every $x \in (V \cup E)$ it holds that $\lambda(x) = \lambda'(b(x))$;
5. for every $(x, pn) \in \mathrm{dom}(\sigma)$ it holds that $\sigma(x, pn) = \sigma'(b(x), pn)$;
6. for every $(x, pn) \notin \mathrm{dom}(\sigma)$ it holds that $(b(x), pn) \notin \mathrm{dom}(\sigma')$.

Note that this notion of equivalence is symmetric; i.e, $g' \equiv g$ if and only if $g \equiv g'$.

Based on this equivalence relationship, we now can define the information preservation property. The following formal definition captures a notion of this property that shall allow us to indicate that specific LPG-to-RDF* mappings have this property only for designated subsets of all possible LPGs.

Definition 4.2 (information preservation of LPG-to-RDF* mappings). Let $\mathcal{P}$ be a set of LPGs. An LPG-to-RDF* mapping m is *information preserving for* $\mathcal{P}$ if there exits a mapping m' from RDF* graphs to LPGs such that i) m' is computable and ii) for every $g \in \mathcal{P}$ it holds that $m'(m(g)) \equiv g$.

We shall show that our proposed LPG-to-RDF* mapping is information preserving for LPGs that do *not* contain multiple distinct edges with the same head, the same tail, and the same label. Hereafter, such LPGs are called *edge-unique*.

Definition 4.3 (edge unique). An LPG $(V, E, \rho, \lambda, \sigma)$ is *edge-unique* if for every pair of edges $e, e' \in E$ with $\rho(e) = \rho(e')$, it holds that $\lambda(e) \neq \lambda(e')$ or $e = e'$.

Example 4.3. While the LPGs illustrated in Figures 1 and 2 are edge-unique, respectively, the LPG in Figure 3 is not edge-unique.

Now we have provided all the preliminaries for our first main technical result:

Proposition 1. *Let $c = (nm, nlm, elm, pm, u_{\mathsf{label}})$ be an LPG-to-RDF* configuration such that $u_{\mathsf{label}} \notin \mathrm{img}(pm)$, where $\mathrm{img}(pm)$ denotes the image of the property name mapping pm. The c-specific direct LPG-to-RDF* mapping $dm^c_{\mathsf{L2R}^\star}$ is information preserving for the set of all LPGs that are edge-unique.*

Proof (sketch). The idea to prove Proposition 1 is to construct an inverse of the mapping $dm^c_{\mathsf{L2R}^\star}$ as required by Definition 4.2. The crux is to use the inverse vm^{-1} of the property value mapping vm (cf. Section 2.3) as well as the inverses of the injective functions nlm, elm and pm. Then, it is not difficult to construct the required inverse mapping by taking into account the properties in Note 1 below.

Note 1. For every LPG $g = (V, E, \rho, \lambda, \sigma)$, the following properties of the resulting RDF* graph $dm^c_{\mathsf{L2R}^\star}(g)$ follow trivially from Definition 3.2.

Property 1. The four subsets G_{nl}, G_{e}, G_{np}, and G_{ep} are pairwise disjoint.

Property 2. If $u_{\mathsf{label}} \notin \text{img}(pm)$, then it is possible to decide for every RDF* triple $t = (s, p, o)$ in the RDF* graph $dm^c_{\mathsf{L2R}^\star}(g)$ whether $t \in G_{\mathsf{nl}}$, $t \in G_{\mathsf{e}}$, $t \in G_{\mathsf{np}}$, or $t \in G_{\mathsf{ep}}$. That is,
- if $p = u_{\mathsf{label}}$, then $t \in G_{\mathsf{nl}}$;
- if $s \in \mathcal{T}^\star$, then $t \in G_{\mathsf{ep}}$;
- if $p \neq u_{\mathsf{label}}$, $s \notin \mathcal{T}^\star$, and $o \in \mathcal{L}$, then $t \in G_{\mathsf{np}}$;
- if $p \neq u_{\mathsf{label}}$, $s \notin \mathcal{T}^\star$, and $o \notin \mathcal{L}$, then $t \in G_{\mathsf{e}}$.

Property 3a. The number of triples in G_{nl} is equivalent to the size of V. Hence, for every node $n \in V$, there exists a unique triple in G_{nl}.

Property 3b. If g is edge-unique, then the number of triples in G_{e} is equivalent to the number of edges $e \in E$ for which there does not exist a $pn \in \mathit{Props}$ such that $(e, pn) \notin \text{dom}(\sigma)$.

Property 3c. The number of triples in G_{np} is equivalent to the number of pairs $(n, pn) \in \text{dom}(\sigma)$ such that $n \in V$.

Property 3d. If g is edge-unique, then the number of triples in G_{ep} is equivalent to the number of pairs $(e, pn) \in \text{dom}(\sigma)$ such that $e \in E$. □

As a final remark we emphasize that, in practice, the limitation of edge uniqueness may not be as limiting as it may seem. Information represented by introducing multiple edges between nodes can also be modeled by using an alternative, more explicit approach. For instance, the relationship captured by each of the multiple edges may be modeled as a separate node.

5 Property Graph-based Query Semantics for SPARQL*

Our proposed LPG-to-RDF* mapping is sufficient as a formal foundation for users who aim to query LPGs using SPARQL* *after first converting the LPGs into RDF* graphs*. To also support use cases in which a conversion into RDF* graphs is not desired or not practical, in this section we define a formal semantics for evaluating SPARQL* queries *directly over LPGs*. Hence, this semantics provides the foundation for implementations in which users express SPARQL* queries in terms of a *virtual RDF* view* of an LPG that is stored in an LPG system.

Our approach to formalize the new LPG-specific query semantics in this section is to first define an evaluation function for triple* patterns only. Thereafter, we define the LPG-specific evaluation function for BGP* queries, which then may easily be extended to full SPARQL* using exactly the same formalization as used in our technical report [5] for the RDF*-based semantics (cf. Section 2.2).

The idea of evaluating a triple* pattern tp over an LPG is to identify RDF* triples that match tp and that exist in the virtual RDF* view of the LPG, where this view is established by the same functions $\mathsf{nl}^{c,g}$, $\mathsf{e}^{c,g}$, $\mathsf{np}^{c,g}$ and $\mathsf{ep}^{c,g}$ that we use to define our LPG-to-RDF* mapping (Definition 3.2). The following definition of an LPG-based evaluation function for triple* patterns captures this idea formally.

Definition 5.1 (evaluation of triple* patterns over LPGs). Let $c = (nm, nlm, elm, pm, u_{\mathsf{label}})$ be an LPG-to-RDF* configuration, $tp = (s, p, o)$ be a triple* pattern, and $g = (V, E, \rho, \lambda, \sigma)$ be an LPG. The *c-specific evaluation of tp over g*, denoted by $[\![tp]\!]^c_g$, is a set of solution* mappings that is defined as

$$[\![tp]\!]^c_g = \Omega_{\mathsf{nl}} \cup \Omega_{\mathsf{e}} \cup \Omega_{\mathsf{np}} \cup \Omega_{\mathsf{ep}}\,, \quad \text{where}$$

$$\Omega_{\mathsf{nl}} = \{\mu \mid \mathrm{dom}(\mu) = \mathrm{vars}(tp) \text{ and there exists } n \in V \text{ s.t. } \mu[tp] = \mathsf{nl}^{c,g}(n)\},$$

$$\Omega_{\mathsf{e}} = \{\mu \mid \mathrm{dom}(\mu) = \mathrm{vars}(tp) \text{ and there exists } e \in E \text{ s.t. } \mu[tp] = \mathsf{e}^{c,g}(e)\},$$

$$\Omega_{\mathsf{np}} = \{\mu \mid \mathrm{dom}(\mu) = \mathrm{vars}(tp) \text{ and there exists } (n, pn) \in \mathrm{dom}(\sigma) \text{ s.t. } n \in V \text{ and } \mu[tp] = \mathsf{np}^{c,g}(n, pn)\},$$

$$\Omega_{\mathsf{ep}} = \{\mu \mid \mathrm{dom}(\mu) = \mathrm{vars}(tp) \text{ and there exists } (e, pn) \in \mathrm{dom}(\sigma) \text{ s.t. } e \in E \text{ and } \mu[tp] = \mathsf{ep}^{c,g}(e, pn)\}.$$

Example 5.1. Let g be the LPG in Figure 1 (as captured formally in Example 2.4). If we use the LPG-to-RDF* configuration of Example 3.1 to evaluate the three triple* patterns $tp_1 = (?x, \mathsf{p{:}name}, ?n)$, $tp_2 = (?y, \mathsf{r{:}influencedBy}, ?z)$, and $tp_3 = (?t, \mathsf{p{:}significance}, ?s)$ over g, we get the following results (with $b_1, b_2 \in \mathcal{B}$):

$$[\![tp_1]\!]^c_g = \{\mu, \mu'\} \text{ where } \mu = \{?x \to b_1, ?n \to \text{"Stanley Kubrick"}\} \text{ and } \mu' = \{?x \to b_2, ?n \to \text{"Oscar Welles"}\},$$

$$[\![tp_2]\!]^c_g = \{\mu\} \text{ where } \mu = \{?y \to b_1, ?z \to b_2\},$$

$$[\![tp_3]\!]^c_g = \{\mu\} \text{ where } \mu = \{?t \to (b_1, \mathsf{r{:}influencedBy}, b_2), ?s \to 0.8\}.$$

We now define the LPG-specific evaluation function for BGP* queries in terms of joins of the results obtained from the triple* patterns in the given BGP*.

Definition 5.2 (LPG-specific evaluation). Let $c = (nm, nlm, elm, pm, u_{\mathsf{label}})$ be an LPG-to-RDF* configuration, $B = \{tp_1, tp_2, \ldots, tp_m\}$ be a BGP*, and $g = (V, E, \rho, \lambda, \sigma)$ be an LPG. The *c-specific evaluation of B over g*, denoted by $[\![B]\!]^c_g$, is a set of solution* mappings that is defined as

$$[\![B]\!]^c_g = [\![tp_1]\!]^c_g \bowtie [\![tp_2]\!]^c_g \bowtie \ldots \bowtie [\![tp_m]\!]^c_g,$$

where the join of two sets Ω and Ω' of solution* mappings is defined as usual [5]:

$$\Omega \bowtie \Omega' = \{\mu \cup \mu' \mid \mu \in \Omega \text{ and } \mu' \in \Omega' \text{ s.t. } \mu \text{ and } \mu' \text{ are compatible [5]}\}.$$

Example 5.2. Consider the following BGP* consisting of three triple* patterns:

$$B = \{\ ((?p1, \mathsf{r{:}influencedBy}, ?p2), \mathsf{p{:}significance}, ?s),\ (?p1, \mathsf{p{:}name}, ?n1),\ (?p2, \mathsf{p{:}name}, ?n2)\ \}.$$

If we use the LPG-to-RDF* configuration of Example 3.1 to evaluate B over the LPG g in Figure 1, we obtain $[\![B]\!]^c_g = \{\mu\}$ where $\mu = \{?p1 \to b_1, ?p2 \to b_2, ?n1 \to \text{"Stanley Kubrick"}, ?n2 \to \text{"Oscar Welles"}, ?s \to 0.8\}$ with $b_1 \in \mathcal{B}$ and $b_2 \in \mathcal{B}$.

A natural question at this point is whether it makes a difference in terms of query results to use our new LPG-specific query semantics directly over an LPG as opposed to converting the LPG into an RDF* graph and then using the query semantics defined for the RDF* data model (as introduced in Section 2.2).

Our second main technical result shows that it makes no difference. More precisely, the following proposition shows that query results under the new LPG-specific query semantics coincide with the query results of the RDF*-specific semantics when applying our LPG-to-RDF* mapping in the context of the latter.

Proposition 2. *For every LPG-to-RDF* configuration c, every BGP* B, and every LPG g, it holds that $[\![B]\!]^c_g = [\![B]\!]_G$ where $dm^c_{\mathsf{L2R}^\star}(g) = G$.*

Proof (sketch). To prove Proposition 2 it is sufficient to show that the semantics coincide for triple* patterns. Then, the fact that the semantics coincide for every BGP* $B = \{tp_1, tp_2, \ldots, tp_m\}$ follows from the following equivalence that holds for every RDF* graph G:

$$[\![B]\!]_G = [\![tp_1]\!]_G \bowtie [\![tp_2]\!]_G \bowtie \ldots \bowtie [\![tp_m]\!]_G. \tag{1}$$

Note that this equivalence can easily be shown based on Definition 2.3 and the definition of the join operation for sets of solution mappings (cf. Definition 5.2).

To show that the semantics coincide for every triple* pattern tp, let c be an arbitrary LPG-to-RDF* configuration, g be an arbitrary LPG, and G be an RDF* graph such that $dm^c_{\mathsf{L2R}^\star}(g) = G$. We have to show that $[\![\{tp\}]\!]^c_g \subseteq [\![\{tp\}]\!]_G$ and $[\![\{tp\}]\!]^c_g \supseteq [\![\{tp\}]\!]_G$. The crux of showing both of these subset relationships is that both the four subsets G_{nl}, G_{e}, G_{np} and G_{ep} in Definition 3.2 and the four subsets Ω_{nl}, Ω_{e}, Ω_{np} and Ω_{ep} in Definition 5.1 are defined in terms of the same functions $\mathsf{nl}^{c,g}$, $\mathsf{e}^{c,g}$, $\mathsf{np}^{c,g}$ and $\mathsf{ep}^{c,g}$. □

6 Closing Remarks

Establishing a precise understanding of formal mappings and query semantics as done in this paper is a necessary foundation to systematically develop solid, practical approaches to make the different graph database models interoperable. As a concrete next step, enabled by the work in this paper, we aim to develop efficient algorithms that apply the proposed mapping to convert LPGs to RDF* data and to use SPARQL* as a common query language.

Acknowledgments. The author's work on this paper has been funded by the CENIIT program at Linköping University (project no. 17.05).

References

1. R. Angles, M. Arenas, P. Barceló, A. Hogan, J. Reutter, and D. Vrgoc. Foundations of Modern Query Languages for Graph Databases. *ACM Comp. Surv.*, 50(5), 2017.
2. R. Cyganiak, D. Wood, M. Lanthaler, G. Klyne, J. J. Carroll, and B. McBride. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, Feb. 2014.
3. S. Harris, A. Seaborne, and E. Prud'hommeaux. SPARQL 1.1 Query Language. W3C Recommendation, Mar. 2013.
4. O. Hartig. Foundations of RDF* and SPARQL* – An Alternative Approach to Statement-Level Metadata in RDF. In *Proc. of the 11th AMW Workshop*, 2017.
5. O. Hartig and B. Thompson. Foundations of an Alternative Approach to Reification in RDF. *CoRR*, abs/1406.3399, 2014. Online: http://arxiv.org/abs/1406.3399.
6. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and Complexity of SPARQL. *ACM Transactions on Database Systems*, 34(3), 2009.
7. I. Robinson, J. Webber, and E. Eifrém. *Graph Databases: New Opportunities for Connected Data.* O'Reilly Media, Inc., 2nd edition, 2015.
8. J. Sequeda, M. Arenas, and D. P. Miranker. On Directly Mapping Relational Databases to RDF and OWL. In *Proc. of the 21st World Wide Web Conf.*, 2012.