Knowledge graph multilevel abstraction: A property graph reification based approach

Selsebil Benelhaj-Sghaier[1], Annabelle Gillet[1], and Éric Leclercq[1]

Laboratoire d'Informatique de Bourgogne - EA 7534
University of Burgundy, Dijon, France
Selsebil_Ben-El-Haj-Sghaier@etu.u-bourgogne.fr
annabelle.gillet@u-bourgogne.fr
eric.leclercq@u-bourgogne.fr

Abstract. Adding knowledge to data or information is an essential step for new information system applications, which need to break down silos in order to define views adapted to evolving needs. Conventional integration approaches have difficulties in meeting the flexibility required by new needs, mainly because their schema is rigid. Graph-based approaches, such as knowledge graphs, are promising, as they allow to use different graph models such as RDF or property graph models. However, they do not make it easy to describe complex relationships at different levels of abstraction. The reification process, already well-studied for RDF, is a promising solution to add new representation capabilities. This paper delves into the process of knowledge reification within the property graph model as a novel approach to enhance the expressivity of knowledge graph model by adding capabilities of representing complex relationships and multilevel abstractions. Based on the study of reification models for RDF, we formalize a new model for property graphs by generalizing the different reification techniques.

Keywords: Knowledge graph · Knowledge reification · Property graph · Multilevel knowledge representation

1 Introduction

As the variety and volume of data sources continue to increase, new challenges in handling data and knowledge emerge. Indeed, these data sources can be represented with various models, each possessing different characteristics and containing information of different nature [5]. In addition, data evolve rapidly, requiring data management tools with a high level of flexibility.

Furthermore, the multiplication of data-centric applications corresponding to new use cases require to break information system silos by defining multiple views of the data [1].

In this context, knowledge graphs [9, 11] have gained increasing interest as a flexible solution to organize and represent knowledge in a structured format. They use a graph-based approach, where entities are represented as nodes and

relationships between them are represented as edges. This structure makes it easy to store, retrieve, and analyze data, and it is particularly well-suited for a wide range of applications [20] such as search engines, recommendation systems, and medical knowledge integration. Besides, the expressivity of knowledge graphs is essential for handling complex real-world relationships and creating precise and detailed knowledge representations.

To improve the expressivity of knowledge graphs in the scope of knowledge representation, knowledge graphs allow the use of rich data models, in term of knowledge representation capabilities, such as the property graph model [2]. However, its expressivity is still limited, and cannot represent complex relationships (e.g., a property that would concern multiple nodes) or abstraction levels (e.g., an abstract node that represents a subgraph).

In this paper, we propose to extend the property graph model with reification for enhancing its expressivity.

We will apply our approach to represent complex relationships among entities and to achieve multilevel abstraction in knowledge representation.

The paper is organised as follows. In section 2, we study the different graph models used in knowledge graphs representation. In section 3 we make a study of related works aiming at improving the expressivity of knowledge graphs using the reification process. In section 4, we propose a formal definition of the reified property graphs and we underline the added value of our model. Finally, section 5 concludes the article and presents future work directions.

2 Knowledge graph models

A knowledge graph is a flexible way for conceptualizing, representing, and integrating diverse and incomplete real-world knowledge [11, 17]. Knowledge graphs can be categorized based on the graph model they use. In this section, we will explore the evolution of knowledge graph models in terms of their expressivity, highlighting limitations in existing models.

A **directed graph** [3] is defined as the triple (V, E, ρ), where V represents a non-empty set of nodes and E constitutes a set of edges connecting nodes in V. $\rho : E \to (V \times V)$ is a total function which gives the couple of nodes associated through an edge.

A **directed edge-labelled graph** (DEL) allows to differentiate multiples edges connecting the same couple of nodes. To do so, it uses labels on edges according to the type of the relationship. A DEL graph is defined as a quadruplet (V, E, ρ, λ), where the additional element $\lambda : E \to \mathcal{L}$ is a total function, with $\mathcal{L}$ being a set of labels.

A **property graph** is a 5-tuple $(V, E, \rho, \lambda, \sigma)$, where $\sigma : (V \cup E) \times Prop \to Val$ is a partial function, where $Prop$ is a finite set of properties and Val is a set of values. If $v \in V$ (resp., $e \in E$), $p \in Prop$, and $\sigma(v, p) = s$ (resp., $\sigma(e, p) = s$), then s is the value of the property p for the node v (resp., the edge e) [2]. Compared to the DEL model, $\lambda : (V \cup E) \to 2^{\mathcal{L}}$ assigns a set of labels for each node and edge. In property graphs, aside from labels, both nodes and

edges have properties in the form of property-value pairs. This proves valuable in providing additional metadata and semantics to the nodes and relationships between them [7].

The **hypergraph** model is a generalization of a graph, able to connect more than two nodes. A hypergraph $H = (V, E)$ is defined as a family of hyperedges E, where each hyperedge is a non-empty subset of V, V being finite set of nodes [4]. It has been used in knowledge graphs to represent relationships among multiple entities [8], such as a diploma, a person and a university linked by a graduation.

Until the rise of knowledge graphs, the DEL model was the most used. However, its expressivity is rather limited, as data properties can only be added as a link labelled with the name of the property and a node representing the value of the property. Thus, from a model point of view, the same semantic is used for data properties and for object properties (that link two entities). The property graph model is an improvement regarding the expressivity, as data properties can be directly added as node properties. Nevertheless, there are still plenty of use cases that cannot be easily represented with the property graph model, such as a relationship among more than two entities or a relationship between two links. The hypergraph model overcomes a part of these limits by allowing to link more than two nodes, but a more general concept is needed to greatly improve the expressivity of knowledge graphs. In this context, the process of reification is a good candidate.

3 Related works

-Definition of Reification
- More specifically, we focus on reification as an abstraction mechanism to enhance the expressivity of graph models.
-In the literature, reification has been applied on RDF graph based on DEL model.
- There are two categories of RDF reification : Triple-based reification and subgraph-based reification.
- First category includes Standard, SP and RDF* reification. the second category is about named graph reification.
- For each type of reification , I define it + ref and I add a recent work.

Reification is the process of objectifying something and treating it as an element of its own right in the domain of discourse. More specifically, we focus on reification as an abstraction mechanism to enhance the expressivity of graph models. In the literature, reification has been used to enhance the expressivity of labelled graph model, namely the RDF graph by associating metadata to RDF triples [15]. There are two categories of RDF reification : Triple-based reification and subgraph-based reification. The first category includes standard, singleton property and RDF* reification. The second category is about named graph reification.

Regarding the first RDF reification category, the **standard RDF reification** [13], expressed in the RDF Primer, refers to the process of representing an RDF statement as a new resource, an instance of the `rdf:statement` class, with the main properties `rdf:subject`, `rdf:predicate` and `rdf:object`. In 2014, Nguyen et al. [14] proposed an alternative approach to RDF reification, primarily based on predicates rather than subjects or objects. This approach is the **singleton property**, which adds a unique predicate to the original predicate of an RDF triple and carries metadata in additional triples related to the unique predicate as a subject. Since the main idea of singleton property reification is to add metadata on the predicate which is an edge connecting two nodes, the work of Rosso *et al.* in 2020, [16] was inspired by this technique of adding metadata to the edges. They proposed to assign to each edge the pair (qualifier, value) representing the name of the metadata and its value. In 2023, the work of Tan *et al.* [18] is also inspired by the same idea of adding metadata to the edges. Indeed, they have represented causal relationships with edges from a Cause node towards an Effect node by adding weight on edges to indicate the strength of the causal effect between nodes. The last type of triple-based reification is the **RDF* reification** [10]. It is as an extension of standard reification to simplify it. It does not require the creation of a resource to reify a triple, thereby eliminating the multiple statements needed for explicit reification. According to RDF*, the subject or the object of an RDF triple could be itself a triple forming a nested triple, defined by W3C as a *quoted triple*. In 2022, Kovacevic *et al.* [12] based their work on the RDF* technique to annotate data triples with temporal metadata. And therefore, the nested triple is timestamped triple. In 2023, Xiong *et al.*[19] proposed nested facts, which involve adding links between links. This approach can be seen as RDF* reification between two triples.

The second category of RDF reification is based on **named graph** [6]. A named graph G is a subset of interconnected nodes associated with a unique identifier. It is represented as a pair (G, ID), where ID is a URI: the identifier of the named graph G. The approach of named graph reification uses the named graph as a context in which the principle triple and its metadata triple(s) occur.

critique /discussion sur les techniques de la reification RDF

The studied related works show a need for extending models of knowledge graphs using RDF reification, in order to integrate more advanced knowledge representation techniques such as metadata representation.

In terms of model expressivity, the standard RDF reification, the singleton property and the RDF* reification are similar. They allow to add information to a triple, the first one by creating a new resource, the second one by directly adding information on the predicate, and the last one by using the triple as is. However, they are not flexible as they require to reify only a triple. The named graph reification technique is less restrictive regarding the selection of elements as it uses a subgraph as input, but it only groups elements and does not include a mechanism to add knowledge about this subgraph.

4 A model for property graph reification

To enhance the expressiveness of the property graph model, as a model covering most of the capabilities of other graph models, we draw inspiration from the principle of RDF reification to propose a model capable of representing different levels of abstraction. In the following section, we explain the basic strategies of property graph reification and then we propose a formal definition.

4.1 Reification strategies on property graph

As the property graph model allows more advanced representations than an RDF graph, these evolutions have to be considered before applying reification techniques on property graphs. First, while data properties and object properties are both represented with an edge in RDF, within a property graph, a data property becomes a node property. Thus, reification on the property graph model must include node properties to keep at least the same level of expressivity. Second, in a property graph, each node and edge can have multiple labels, rather than just one in a RDF graph. They can also replace some edges, as for example to define multiple classes for a given node. Therefore, the reification mechanism must consider the labels for nodes and edges.

For the improvement we are going to make regarding RDF reification, we propose a property graph reification enables the representation of a sub-graph in the form of a node, which acts itself as a standard node in the graph. The sub-graph is a set of nodes (at least two nodes), edges, properties, and labels. This mechanism can potentially be recursive, thereby enabling the modeling of different levels of abstraction.

4.2 Formal model for property graph reification

To define reification on property graphs, we start from the formal definition of the property graph model and extend it.

Therefore, a property graph model G with reification is defined by a tuple

$$G = (V, E, R, \rho, \lambda, \sigma, \alpha)$$

where :

- $R \subseteq V$, R is a subset of V containing the reified nodes;
- $\alpha : R \to SG$ is a total function mapping the reified nodes to their corresponding subgraph. SG is a finite set of tuples $sg = (V_r, E_r, R_r, \rho_r, \lambda_r, \sigma_r, \alpha_r)$.
  If $r \in R$ and $sg \in SG$, then $\alpha(r) = sg$ where sg is the subgraph of the reified node r.
  In the definition of sg, $V_r \subseteq V$, $E_r \subseteq E$, $R_r \subseteq R$, $\rho_r = \rho|_{E_r}$, $\lambda_r = \lambda|_{V_r \cup E_r}$ and $\lambda_r : (V_r \cup E_r) \to 2^{\mathcal{L}_r}$, $\sigma_r = \sigma|_{(V_r \cup E_r) \times Prop_r}$, $\alpha_r = \alpha|_{R_r}$. The functions $\rho_r, \lambda_r, \sigma_r$ and α_r are restricted functions. $Prop_r$ is a finite set of selected properties of the subgraph and $\mathcal{L}_r$ is a finite set of selected labels, thus $Prop_r \subseteq Prop$ and $\mathcal{L}_r \subseteq \mathcal{L}$.

The expressivity of knowledge graph is up to the knowledge representation capabilities of the used graph model. Our reification approach applied to property graphs participate into improving the expressivity of property graph model by enabling the following representation capabilities :

- Abstraction: By selecting a subset of nodes along with their labels, properties and edges, we can effectively abstract them and represent them with a new node corresponding to a subgraph $sg \in SG$.

- Multi-level abstraction: The abstraction mechanism is recursive: a reified node can also represent other reified nodes.

- Abstraction annotation: A reified node $r \in R$ benefits from all the modelling capabilities of standard nodes as $R \subseteq V$. It allows to define labels and properties for reified nodes, as well as edges whether the destination node is a standard or a reified node.

4.3 Reified node specification

To build a reified node, users must specify which elements are concerned by the reification. As stated by the model definition, the model allows to reify on nodes, links, labels and properties. Thus, the building function must accept these elements individually or any combination of them.

To do so, we propose the following function:

$$\beta(V_r, E_r, \lambda_r, \sigma_r) = r$$

where V_r is the set of selected nodes, E_r the set of selected edges, $\lambda_r : (V_r \cup E_r) \rightarrow 2^{\mathcal{L}_r}$ restricts the labels kept, and $\sigma_r : (V_r \cup E_r) \times Prop_r \rightarrow Val$ restricts the properties kept.

As λ_r and σ_r have their domain restricted on the selected nodes and edges, it guarantees that the selected labels and properties belong to an existing node or edge of the reification. Furthermore, it is a flexible mechanism as it allows to keep a given property or label for a specific node or edge but not necessarily for all the nodes and edges that have this property or label. $\rho_r = \rho|_{E_r}$ is built automatically from E_r, by restricting the function ρ on E_r.

To obtain R_r stated in the model, we rely on R of the global graph, as reified nodes of a subgraph are also reified nodes in the graph containing the subgraph. Therefore, $R_r = \{v | v \in R \cap V_r\}$.

5 Conclusion and perspectives

Representing real-world data using knowledge graphs often requires complex relationships. The property graph model stands out as a particularly concise approach compared to other graph models for representing knowledge. Its strength

lies in its ability to represent knowledge with a reduced structural size using labels and properties on nodes and edges. However, this model does have its limitations, notably its inability to represent complex relationships between nodes and its inability to define multiple levels of abstraction. To overcome these limitations, we extend the current property graph model and propose a new model supporting reification. We have formalised our reification approach that empowers the property graph model to model complex relationships by representing a sub-graph of a set of nodes, edges as well as their related properties and labels in the form of a node. Furthermore, this mechanism is recursive and enables multilevel knowledge abstraction. In addition it offers the possibility of adding metadata to a sub-graph since it support abstraction annotation.

Our forthcoming work is structured around two main axes. Firstly, we plan to proceed with the technical implementation of our approach. Secondly, we intend to adapt current property graph query languages to our model. This adaptation is crucial to enable effective management of multi-level abstraction when querying knowledge graphs. For example, queries will need to be capable of aggregating properties, particularly when calculating a property based on the properties of the constituent elements of an abstract node. These two axes represent a crucial step in fully exploiting the potential of our model in practical contexts and facilitating its integration into existing applications and computer systems.

References

1. Abiteboul, S., Arenas, M., Barceló, P., Bienvenu, M., Calvanese, D., David, C., Hull, R., Hullermeier, E., Kimelfeld, B., Libkin, L., et al.: Research directions for principles of data management. Dagstuhl Manifestos **7**(1), 1–29 (2018)
2. Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J., Vrgoč, D.: Foundations of modern query languages for graph databases. ACM Computing Surveys (CSUR) **50**(5), 1–40 (2017)
3. Bang-Jensen, J., Gutin, G.: Classes of directed graphs, vol. 11. Springer (2018)
4. Berge, C.: Graphs and hypergraphs, volume 6 of math. Library. North-Holland, Amsterdam (1973)
5. Bernstein, P.A., Halevy, A.Y., Pottinger, R.A.: A vision for management of complex models. ACM Sigmod Record **29**(4), 55–63 (2000)
6. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs. Journal of Web Semantics **3**(4), 247–267 (2005)
7. Das, S., Srinivasan, J., Perry, M., Chong, E.I., Banerjee, J.: A Tale of Two Graphs: Property Graphs as RDF in Oracle. In: International Conference on Extending Database Technology (EDBT). pp. 762–773 (2014)
8. Fatemi, B., Taslakian, P., Vazquez, D., Poole, D.: Knowledge hypergraphs: Prediction beyond binary relations. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20. pp. 2191–2197. International Joint Conferences on Artificial Intelligence Organization (7 2020). https://doi.org/10.24963/ijcai.2020/303, https://doi.org/10.24963/ijcai.2020/303, main track
9. Gutiérrez, C., Sequeda, J.F.: Knowledge graphs. Communications of the ACM **64**(3), 96–104 (2021)

10. Hartig, O.: Foundations of RDF* and SPARQL*:(an alternative approach to statement-level metadata in RDF). In: AMW 2017 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017. vol. 1912. Juan Reutter, Divesh Srivastava (2017)
11. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., et al.: Knowledge graphs. ACM Computing Surveys (CSUR) **54**(4), 1–37 (2021)
12. Kovacevic, F., Ekaputra, F.J., Miksa, T., Rauber, A.: Starvers-versioning and timestamping rdf data by means of rdf*-an approach based on annotated triples (2022)
13. Manola, F., Miller, E.: Rdf reification (2004), https://www.w3.org/TR/rdf-primer/#reification
14. Nguyen, V., Bodenreider, O., Sheth, A.: Don't like RDF reification? making statements about statements using singleton property. In: Proceedings of the 23rd international conference on World wide web. pp. 759–770 (2014)
15. Orlandi, F., Graux, D., O'Sullivan, D.: Benchmarking RDF metadata representations: Reification, singleton property and RDF. In: 2021 IEEE 15th International Conference on Semantic Computing (ICSC). pp. 233–240. IEEE (2021)
16. Rosso, P., Yang, D., Cudré-Mauroux, P.: Beyond triplets: hyper-relational knowledge graph embedding for link prediction. In: Proceedings of the web conference 2020. pp. 1885–1896 (2020)
17. Sequeda, J., Lassila, O.: Building enterprise knowledge graphs. In: Designing and Building Enterprise Knowledge Graphs, pp. 97–128. Springer (2021)
18. Tan, F.A., Paul, D., Yamaura, S., Koji, M., Ng, S.K.: Constructing and interpreting causal knowledge graphs from news. arXiv preprint arXiv:2305.09359 (2023)
19. Xiong, B., Nayyeri, M., Luo, L., Wang, Z., Pan, S., Staab, S.: Neste: Modeling nested relational structures for knowledge graph reasoning. arXiv preprint arXiv:2312.09219 (2023)
20. Zou, X.: A survey on application of knowledge graph. In: Journal of Physics: Conference Series. vol. 1487, p. 012016. IOP Publishing (2020)