Formal Concept Analysis for Semantic Compression of Knowledge Graph Versions

Damien Graux[1,2], Diego Collarana[3,4], Fabrizio Orlandi[2]

[1] Inria, Université Côte d'Azur, CNRS, I3S, France
[2] ADAPT SFI Centre, Trinity College Dublin, Ireland
[3] Fraunhofer IAIS, Sankt Augustin, Germany
[4] Universidad Privada Boliviana, Bolivia
damien.graux@inria.fr, orlandif@tcd.ie,
diego.collarana.vargas@iais.fraunhofer.de

Abstract. Recent years have witnessed the increase of openly available knowledge graphs online. These graphs are often structured according to the W3C semantic web standard RDF. With this availability of information comes the challenge of coping with dataset versions as information may change in time and therefore deprecates the former knowledge graph. Several solutions have been proposed to deal with data versioning, mainly based on computing data deltas and having an incremental approach to keep track of the version history. In this article, we describe a novel method that relies on aggregating graph versions to obtain one single complete graph. Our solution semantically compresses similar and common edges together to obtain a final graph smaller than the sum of the distinct versioned ones. Technically, our method takes advantage of FCA to match graph elements together. We also describe how this compressed graph can be queried without being unzipped, using standard methods.

1 Introduction

Knowledge Graphs (KG) are becoming the preferred data model for integrating heterogeneous data into actionable knowledge. General domain knowledge graphs such as Wikidata [21] and DBpedia [16] have been used as core knowledge sources to develop intelligent applications. Moreover, domain-specific knowledge graphs are being constructed in almost all domains. Knowledge graphs provide a flexible data model allowing the addition and deletion of facts in the graph. Therefore, KGs are dynamic and evolve over time: facts are either added or removed. Such a paradigm leads to the availability of several versions of the *same* KG *e.g.* each version may correspond to a specific release published by the data providers.

Practically, KGs are often modeled according to the RDF standard, proposed by the W3C. In a nutshell, the RDF[1] data model implements multi-relational directed labelled graphs using triples. Indeed, a triple $t = (subject, predicate, object)$ encodes a fact, *e.g.*, `ex:CR7 ex:born ex:Madeira` states that Cristiano Ronaldo was born in Madeira (Figure 1). To efficiently handle the continuously growing knowledge graphs, there is a need for efficient compression techniques to allow practitioners to share and

[1] https://www.w3.org/TR/rdf11-primer/

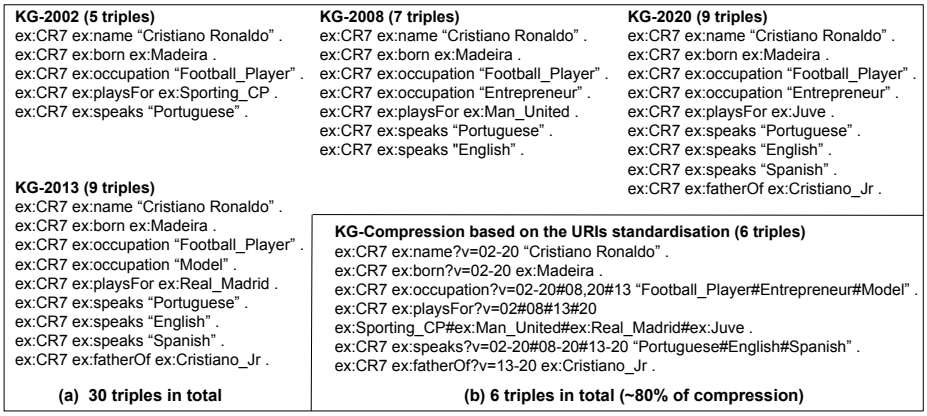

Fig. 1: **Motivating example:** (a) shows four different KG deltas of CR7 entity containing 30 triples in total. (b) depicts our vision of a semantically compressed KG with six triples gaining 80%. The most critical challenge is searching for implicational dependencies in the different deltas. Hence, FCA plays a crucial role in our approach.

store their graphs more easily. Ideally, knowledge graph compression algorithms should serialize RDF data compacting RDF representation in a manner that still allows for querying. By doing so, it should then be possible to query directly compressed graphs without having to “unzip” them prior; this strategy would thereby save memory.

To date, most RDF compression approaches focus on syntactic compression, with systems that modify the standard RDF data model. These systems require the implementation of complex encoders and decoders to deal with various KG versions, computing deltas of triples to capture changes spanning across several versions. For example, Álvarez-García *et al.* [2] implement algorithms directly in the storage solution. The authors apply compact tree structures to the well-known vertical-partitioning technique reaching a great compression degree. However, additional data structures are needed, including a mapping dictionary and adjacency matrices. In this work, we focus on a semantic compression of a set of RDF KGs, *i.e.*, reducing the number of triples by replacing or grouping repetitive parts observed in the various graphs of the set. Technically, our method takes advantage of *formal concept analysis* (FCA) to match graph elements together. Moreover, our method allows to aggregate together concepts considered as “similar” according to a chosen similarity metric.

2 CR7’s career as a motivating example

First, let us take as example four different versions of the same small knowledge graph (Figure 1) encoding facts about Cristiano Ronaldo (*URI = ex:CR7*). These versions provide facts about CR7 at various moments of his career: in 2002, 2008, 2013 & 2020.

We notice that there are redundant facts among the knowledge graph versions, *e.g.*, `ex:CR7 ex:name "Cristiano Ronaldo"` is present in all of them. Intuitively, a compressed graph of these four versions should carry only once this specific state-

ment, mentioning that it is present in each of the considered versions. Such mentions could be done by enriching the URIs of both predicates and objects, specifying e.g. the range of the version where the statement holds. Similarly, the `ex:playsFor` concept changes across versions as Cristiano played for a different team in each version. Our semantic compression, using the same kind of URI annotation, encompasses such changes to wrap all these statements within a single triple (cf. the 4th triple of Figure 1-b).

At the end of this process, the four versions of the CR7 graph are compressed into 6 triples. Moreover, the overall information contained in the obtained compressed KG is strictly the same as the sum of the information available in the various distinct versions. Therefore, for this basic example, our approach allows practitioners to carry one small KG of 6 triples instead of 4 distinct KGs gathering 30 triples.

3 Definitions

In this section, we introduce the main concepts used for our description of the approach and formally define them adapting existing definitions from the literature. First we define the concepts related to RDF Knowledge Graphs and then those related to FCA.

3.1 Knowledge Graph

Definition 1. ***Sets:*** *Let U and L be the mutually disjoint sets of URI references and literals, respectively. Let $P \subseteq U$ be the set of all properties.*

Definition 2. ***RDF triple:*** *An RDF triple $t = (s, p, o) \in U \times P \times (U \cup L)$ displays the statement that the subject s is related to the object o via the predicate p. In this work, we do not consider blank nodes as speficied in the W3C RDF standard.*

Definition 3. ***RDF Knowledge Graph:*** *An RDF Knowledge Graph G is a finite set of RDF triples, where $t = (s, p, o) \in G$. An RDF graph can also be viewed as a finite set of edges t, of the form $s \xrightarrow{p} o$, in a directed edge-labelled graph.*

3.2 Formal Concept Analysis

Formal concept analysis (FCA) is a methodology for extracting a concept hierarchy from sets of entities and their properties [22]. In other words, FCA is based on extracting *formal concepts* from *formal contexts*. Adapting the definitions in [8,14]:

Definition 4. ***Formal Context:*** *A formal context is a triple $X = (E, A, I)$, where E is a set of entities, A is a set of attributes, and $I \subseteq E \times A$ is the incidence: a set of pairs such that $(e, a) \in I$ if and only if the attribute a is defined for entity e.*

Definition 5. ***Formal Concept:*** *Let $X = (E, A, I)$ be a formal context; for a subset of entities $F \subset E$, let $H(F) := \{a \in A \mid \forall f \in F : (f, a) \in I\}$, conversely, for a subset of attributes $G \subset A$, let $K(G) := \{e \in E \mid \forall g \in G : (e, g) \in I\}$. A formal concept of the formal context X is an ordered pair (F, G) such that $H(F) = G$ and $K(G) = F$. If (F, G) and $(F1, G1)$ are formal concepts of X, then $(F, G) \leq (F1, G1)$ if $F \subset F1$ or, equivalently, if $G1 \subset G$.*

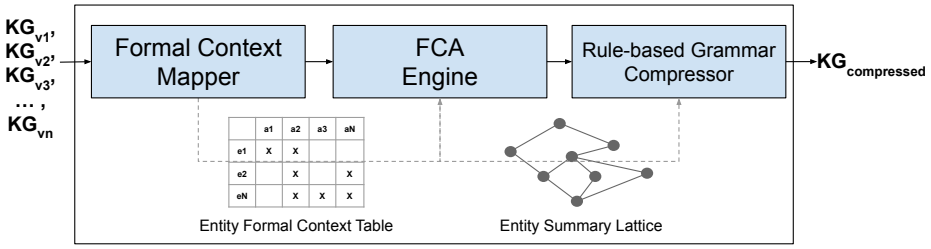

Fig. 2: **Zip function**: our compression approach creates a formal context from the same entities in different KG deltas. Then, we apply FCA to obtain an entity summary lattice. Finally, we execute a rule-based grammar to produce a compressed KG as output.

Definition 6. ***Concept Lattice:*** *Let $X = (E, A, I)$ be a formal context. The set of all formal concepts of X with the partial ordering defined in Definition 5 is called the concept lattice of X.*

4 An approach for zipping Knowledge Graph versions together

Grounded on FCA, we propose a zip function for compressing RDF knowledge graphs providing a solution to the problem of semantically compressing RDF graphs. Figure 2 depicts the main steps defined for our zip function. Our zip function follows a three-fold approach. We receive as input a set of KG versions *i.e.* several *complete* versions of the same Knowledge Graph (containing thereby redundant triples). First, we compute and prepare the formal context. Then, we run an FCA Engine to produce the formal concepts. Finally, we apply a set grammar rules to synthesize a (single) semantically compressed KG from the initial set of graphs.

Formal Context Mapper: First, from the set of KG deltas, we create a formal context, as defined in Definition 4. As E we consider objects of the same type in the KG deltas to be the entities. As A we consider all the properties in E to be the attributes, and the incidence I is given by the use of that property as a predicate on the given subject. Table 1 presents the entity formal context for the example introduced in Section 2, for instance, the RDF triple "`ex:CR7 ex:playsFor ex:Juve`" is only stated in 'KG-2020' as marked in the bottom-right corner cell.

FCA Engine: Second, we apply an FCA implementation to compute the formal concepts out of the formal context as defined in Definition 5. More visually, Figure 3 provides a concept lattice (Definition 6) corresponding to the Section 2 example. For instance at a glance one may see that "`name-CR`, `born-Madeira`, `occu-Player`, `speak-Por`" are statements made in every versions of the Knowledge Graph (practically it would be useful to store only once this information instead of four times).

Rule-based Grammar Compressor: Taking the formal concepts output by the FCA Engine, the idea to obtain a single compressed Knowledge Graph is to "*group*" the redundancies by subjects: meaning that for each distinct subject of the versions we establish the list of (predicate,object) available and then we tag the predicates and the object using the version name. In practice, we take advantage of the fact that the URIs

	name-CR	born-Madeira	occu-Player	plays-Sporting	speaks-Por	occu-Enter	plays-MU	speaks-En	occu-Model	plays-Madrid	speaks-Es	father-CRJ	plays-Juve
CR7-02	x	x	x	x	x								
CR7-08	x	x	x		x	x	x	x					
CR7-13	x	x	x		x			x	x	x	x	x	
CR7-20	x	x	x		x	x		x			x	x	x

Table 1: Example of the entity formal context that our approach creates for CR7 entity.

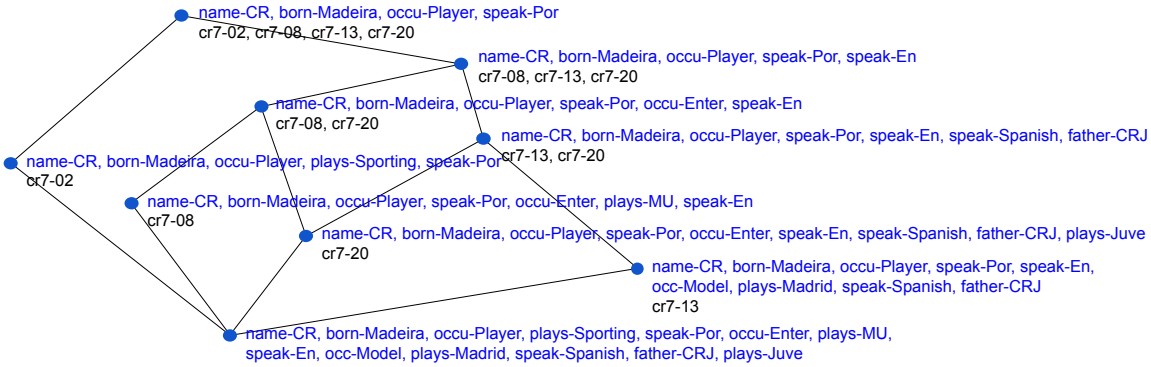

Fig. 3: Example of a resulting lattice after applying FCA to an entity formal context. We browse this lattice, employing rules to compress facts into a compressed KG.

used in RDF can be enriched, and we apply (on the predicates and the objects) the following rule-based grammar to compress the KGs semantically.

- **Hash** "#" is used for ordered separation of version numbers and their corresponding objects, both in new predicate IRIs and new concatenated object IRIs;
- **Hyphen** "–" indicates a continuous range of versions (*i.e.* from-to);
- **Comma** "**,**" indicates a discrete list of individual versions.

For example, in Figure 1, "`ex:CR7 ex:born?v=02-20 ex:Madeira`" means that "`ex:CR7 ex:born ex:Madeira`" was present in all the versions from '02' to '20'.

5 Querying the compressed Knowledge Graph

Fernández et al. [11] categorised all possible retrieval needs for versioned RDF archives. They identify six different types of retrieval needs, regarding the query type (materialisation or structured queries) and the target (version/delta) of the query. These types are listed below, including example queries based on our CR7 example (Figure 1).

1. **Single-version structured queries** are performed only on one specific version.
   `Q1-a`: In 'KG-2013' what team did CR7 play for?
   `Q1-b`: What "occupations" did CR7 have in 'KG-2013'?
2. **Cross-version structured queries** must be satisfied across different versions.
   `Q2-a`: In which KG version did CR7 play for ex:Juve ?
   `Q2-b`: In which KG versions did CR7 have the "occupation" of 'Entrepreneur'?
   `Q2-c`: Which predicates connect CR7 to ex:Madeira and in which versions?

3. **Single-delta structured queries** are the counterparts of the above version-focused queries, but must be satisfied on change instances instead.
   `Q3-a`: What triple with subject 'CR7' and predicate 'ex:speaks' was added between the two consecutive versions 'KG-2002' and 'KG-2008'?
4. **Cross-delta structured queries** are the counterparts of the above version-focused queries, but must be satisfied on change instances instead.
   `Q4-a`: What has changed between the non-consecutive versions 'KG-2002' and '2020'?
5. **Delta materialisation queries** retrieve the delta between two or more versions.
   `Q5-a`: What has changed between the consecutive versions 'KG-2013' and '2020'?
6. **Version materialisation queries** correspond to the retrieval of a full version.
   `Q6-a`: What are all the statements about CR7 valid in version 'KG-2008'?

Naively, these retrieval needs can be satisfied unzipping the KG to re-obtain the different versions. Nevertheless, one advantage of our approach lies in the expressive power of the *de facto* RDF query language: SPARQL[2]. Indeed, practitioners can express each of the aforementioned queries using one single SPARQL query involving filters based on `regular expressions` to grasp the relevant predicates. This therefore allows any standard-compliant triplestore to load and query the compressed KG. For instance, the aforementioned `Q6-a` could correspond to the following SPARQL query:[3]

```
SELECT DISTINCT ?s ?p ?o
WHERE{
  {
    ?s ?p ?o .
    FILTER regex(str(?p), "[?&]v=([^&]*)08.*$").
  }UNION{
    ?s ?p ?o .
    BIND (REPLACE(str(?p), "(..)*-.*", "$1") AS ?strFrom).
    BIND (REPLACE(str(?p), ".*-(..).*", "$1") AS ?strTo).
    FILTER (xsd:int(?strFrom) < 8).
    FILTER (xsd:int(?strTo) > 8).
  }
}
```

Different string functions, from the SPARQL 1.1 standard, are operated in order to check if the predicate `?p` was present in 'KG-2008'. Technically, this is done using regular expressions, for instance above, regex are used to extract the starting and ending versions in case the sought version is "*hidden*" within an interval using a hyphen. As a consequence, the compressed KG can be used to deal with version-related needs together with *conventional* querying while being kept "light" in terms of triples.

6 Related Work

We now provide an overview of the most pertinent related efforts in the areas of FCA for KGs, compression approaches for KGs, and KG versioning.

FCA on KGs: We see FCA supporting different tasks, including: Data Integration using ontologies [13], Entity Matching [20], Entity Temporal Evolution [19] and Modelling Dynamics [14], or Knowledge Exploration where FCA helps assess the completeness of Linked Datasets by mining definitions from RDF annotations [1]. FCA

[2] `https://www.w3.org/TR/sparql11-overview/`

[3] The presented query is simplified for space reasons; it might not generalize to all possible cases, but it could be adapted using additional `FILTER`s like the ones shown. See `https://github.com/badmotor/FCACompressRDF` for test data and all query examples.

is also used in specific cases such as in [4] to propose an alternative semantics for `owl:sameAs`, or to verbalize KG evolution [3]. Similarly, Aquin & Motta describe how to extract relevant questions to an RDF dataset [7]. Furthermore, Formica [12] and Rouane-Hacene et al. [17] extend FCA to respectively support formal ontology constructions in presence of uncertain data and to process multi-relational datasets.

KG Compression Techniques: In terms of knowledge graph compression techniques, several paradigms have been explored. In [9], the authors list the basic and naive techniques to compress an RDF graph. Later, solutions involving pre-processing and rewriting of the graph were proposed: for instance, the HDT representation of RDF triples was suggested [10]. Others describe methods to search the KGs for frequent patterns to factorise them [15]. Additionally, some solutions[4] summarise the KG hence compressing it, *e.g.* [23] compresses parts of a KG considering a set of user-selected queries.

KG Versioning: The literature has been focused on designing systems able to deal with many knowledge graph versions by computing deltas of triples *e.g.* OSTRICH [18]. Closer to our strategy, Cuevas & Hogan explored solutions for representing archives of versioned RDF data using the SPARQL standard [6].

7 Conclusion and Future Work

In this article, we described an architecture to enable the compression of a set of knowledge graph versions into a compressed one, while guaranteeing no loss of information. Furthermore, we presented how such a compressed knowledge graph can be queried directly without decompression, using any existing SPARQL-compliant endpoint.

To strengthen our solution, we will identify and evaluate more robust characters as delimiters for the version parameters and the concatenated object IRIs. This is because the "#" character could be present also in the original (non-concatenated) IRIs, and the "," and "−" characters could break our parser when parsing the version numbers embedded in the predicates. Potentially, the proposed approach generates a high number of unique predicates and unique objects. This could create some performance issues at query time if the data is loaded into a triplestore, as these engines are not usually optimised for such conditions (the RDF Singleton Property model also suffers from the same issue). In the near future, we could allow the configuration of the threshold for the similarity metric and thereby open the discussion towards uncertain data as the zipped KG could be carrying triples whose subjects were considered equal. Finally, an evaluation on real and large datasets will be conducted.

Acknowledgements: We acknowledge the support of the EU H2020 Projects Opertus Mundi (GA 870228), LAMBDA (GA 809965), the EDGE Marie Skłodowska-Curie grant (No. 713567) at the ADAPT SFI Research Centre at Trinity College Dublin (co-funded under the ERDF Grant #13/RC/2106_P2), and the Federal Ministry for Economic Affairs and Energy (BMWi) project SPEAKER (FKZ 01MK20011A).

[4] See [5] for a survey on summarizing semantic graphs.

References

1. Alam, M., Buzmakov, A., Codocedo, V., Napoli, A.: Mining definitions from RDF annotations using formal concept analysis. In: IJCAI, AAAI Press (2015) 823–829
2. Álvarez-García, S., Brisaboa, N.R., Fernández, J.D., Martínez-Prieto, M.A.: Compressed k^2-triples for full-in-memory RDF engines. In: AMCIS. (2011)
3. Arispe, M., Tasnim, M., Graux, D., Orlandi, F., Collarana, D.: Verbalizing the evolution of knowledge graphs with formal concept analysis. In: NLIWOD colocated with ISWC. (2020)
4. Beek, W., Schlobach, S., van Harmelen, F.: A contextualised semantics for `owl:sameAs`. In: ESWC. Volume 9678 of Lecture Notes in Computer Science., Springer (2016) 405–419
5. Čebirić, Š., Goasdoué, F., Kondylakis, H., Kotzinos, D., Manolescu, I., Troullinou, G., Zneika, M.: Summarizing semantic graphs: a survey. VLDB journal **28**(3) (2019) 295–327
6. Cuevas, I., Hogan, A.: Versioned queries over RDF archives: All you need is SPARQL? In: MEPDaW @ ISWC. (2020) 43–52
7. d'Aquin, M., Motta, E.: Extracting relevant questions to an RDF dataset using formal concept analysis. In: K-CAP, ACM (2011) 121–128
8. Denniston, J.T., Melton, A., Rodabaugh, S.E.: Formal Contexts, Formal Concept Analysis, and Galois Connections. Electronic Proceedings in Theoretical Computer Science **129** (2013) 105–120
9. Fernández, J.D., Gutierrez, C., Martínez-Prieto, M.A.: RDF compression: basic approaches. In: WWW. (2010) 1091–1092
10. Fernández, J.D., Martínez-Prieto, M.A., Gutierrez, C.: Compact representation of large RDF data sets for publishing and exchange. In: ISWC, Springer (2010) 193–208
11. Fernández, J.D., Umbrich, J., Polleres, A., Knuth, M.: Evaluating query and storage strategies for RDF archives. Semantic Web **10**(2) (2019) 247–291
12. Formica, A.: Semantic web search based on rough sets and fuzzy formal concept analysis. Knowl. Based Syst. **26** (2012) 40–47
13. Fu, G.: FCA based ontology development for data integration. Inf. Process. Manag. **52**(5) (2016) 765–782
14. González, L., Hogan, A.: Modelling dynamics in semantic web knowledge graphs with formal concept analysis. In: WWW, ACM (2018) 1175–1184
15. Karim, F., Vidal, M.E., Auer, S.: Compacting frequent star patterns in RDF graphs. Journal of Intelligent Information Systems **55**(3) (2020) 561–585
16. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - A large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web **6**(2) (2015) 167–195
17. Rouane-Hacene, M., Huchard, M., Napoli, A., Valtchev, P.: Relational concept analysis: mining concept lattices from multi-relational data. Ann. Math. Artif. Intell. **67**(1) (2013)
18. Taelman, R., Vander Sande, M., Verborgh, R.: OSTRICH: versioned random-access triple store. In: Companion of WWW. (2018) 127–130
19. Tasnim, M., Collarana, D., Graux, D., Orlandi, F., Vidal, M.: Summarizing entity temporal evolution in knowledge graphs. In: Companion volume of WWW, ACM (2019) 961–965
20. Tasnim, M., Collarana, D., Graux, D., Vidal, M.: Context-based entity matching for big data. In: Knowledge Graphs and Big Data Processing. Springer (2020)
21. Vrandecic, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10) (2014) 78–85
22. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: International conference on formal concept analysis, Springer (2009) 314–339
23. Zhang, H., Duan, Y., Yuan, X., Zhang, Y.: ASSG: Adaptive structural summary for RDF graph data. In: ISWC (Posters & Demos). (2014) 233–236