Exploring the Application of Graph-FCA to the Problem of Knowledge Graph Alignment

Sébastien Ferré*

Univ Rennes, CNRS, IRISA
Campus de Beaulieu, 35042 Rennes, France
Email: ferre@irisa.fr, ORCID: 0000-0002-6302-2333

Abstract. Knowledge Graphs (KG) have become a widespread knowledge representation. When different KGs exist for some domain, it is valuable to merge them into a richer KG. This is known as the problem of KG alignement, which encompasses related problems such as entity alignement or ontology matching. Although most recent approaches rely on supervised representation learning, Formal Concept Analysis (FCA) has also been proposed as a basis for symbolic and unsupervised approaches. We here explore the application of Graph-FCA, an extension of FCA for KGs, to different scenarios of KG alignments: (A) when the two KGs have common values, and (B) when pre-aligned pairs are known. We show that, compared to previous FCA-based approaches, Graph-FCA allows for a more natural and scalable representation of the KGs to be aligned, and makes it simpler to extract alignments from the concepts. It also features flexibility w.r.t. different alignment scenarios.

1 Introduction

A knowledge graph (KG) is a modelling of some domain of interest that supports many tasks such as question answering, reasoning over data or machine learning. They are commonly represented in languages of the Semantic Web (RDF(S), OWL), and published on the web as Linked Open Data (LOD) [9]. A KG is a set of entities described by their classes, properties, and relations with other entities. Due to the open nature of the web, it is common that a same domain of interest is modelled by multiple KGs. Those have in general overlapping, hence redundant, information but also complementary information. It is therefore valuable to merge the different KGs that model a same domain into a single richer KG. The difficulty is that different KGs generally use different entity identifiers (URIs) for the same real entities, and even different schemas or ontologies, i.e. different classes, properties and relations.

The problem of merging two KGs consists in discovering equivalence links between entities and/or between schemas, which is known as *knowledge graph alignment*. A number of other terms are used in the literature, partially depending on the focus on entities or on the schema: *entity alignment*, *entity matching*,

* This research is supported by ANR project SmartFCA (ANR-21-CE23-0023).

ontology alignment/matching, *data interlinking* [2,14]. In order to align two KGs, they must have something in common, some alignment seeds. Those seeds can be: a common language for the string values (e.g., names, titles), a set pre-aligned pairs, or a combination of those. Most existing work consider a single scenario, and may not apply to other scenarios. A common scenario (called *Scenario A* in the following) is to only assume a common language, and no pre-aligned pairs. This is an unsupervised setting. Another common scenario in recent literature (called *Scenario B* in the following), assumes that a set of pre-aligned pairs is available [14]. This is a supervised setting where the pre-aligned pairs serve as a training set. Most approaches use similarity measures that combine terminological features (i.e., strings for names, titles, ...), and structural features (i.e., relationships between entities). A recent trend is to use KG embeddings as such features. Formal Concept Analysis (FCA) [7] and FCA extensions, such as Pattern Structures [6] and Relational Concept Analysis [11], have been recently proposed for tasks related to KG alignment [15,3,1]. Formal concepts are used there as a symbolic form of similarity.

In this paper, we explore the application of Graph-FCA [5], an extension of FCA specifically designed for analyzing KGs, to the two main scenarios of KG alignment. We propose an approach that makes three contributions to previous FCA-based approaches.

1. A more natural and scalable representation. The formal context is the *union* (rather than the *product*) of the two KGs to be aligned, with minor adjustments depending on the alignment scenario.
2. A uniform alignment extraction methods for both entities and schema elements (classes, properties and relations). All alignement pairs are extracted from concept extents (rather than from both extents and intents).
3. A flexible approach w.r.t. different scenarios. They are handled through minor adjustments to the representation of KGs.

We show the effectiveness of our approach on a few use cases taken or adapted from previous FCA-based work.

Section 2 states the KG alignment problem. Section 3 presents the related work. Section 4 shortly describes Graph-FCA. Section 5 explains our approach from principles to concrete application, and discusses the results on the two main scenarios. Section 6 concludes and draws perspectives.

2 Problem Statement

We start by defining knowledge graphs (KG), taking care to distinguish between classes, properties and relations in order to allow for a more accurate modelling. Many work on KGs only consider entities and relations.

Definition 1 (knowledge graph). *A* knowledge graph *is a structure* $G = \langle E, C, P, R, T \rangle$*, where* E *is the set of* entities*,* C *is the set of* classes*,* P *is the set of* properties*,* R *is the set of* relations*, and* T *is a set of* triples*. The*

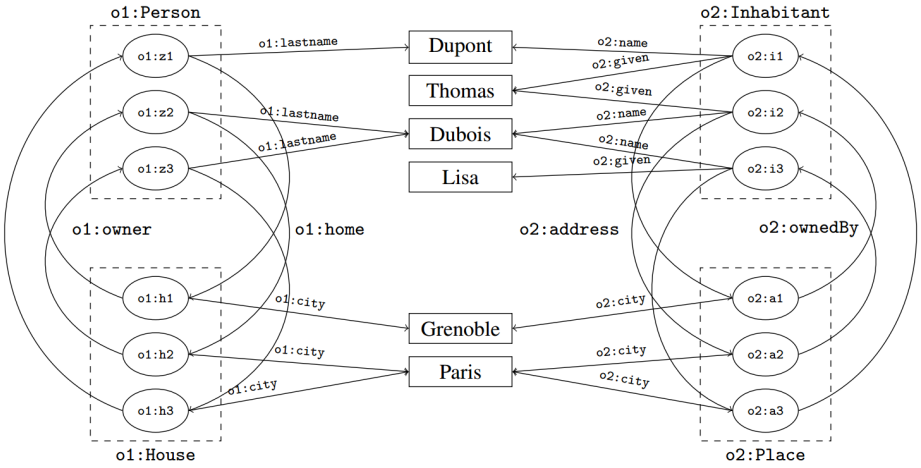


Fig. 1. Two KGs (left and right) with cyclic dependencies (from Atencia *et al* [3])

schema *of the KG is made of the classes, properties and relations. There are three kinds of triples, for any* $e, e' \in E, c \in C, p \in P, r \in R$*:*

- $(e, \texttt{type}, c)$ *states that class* c *is the type of entity* e*;*
- (e, p, v) *states that property* p *has value* v *on entity* e*, where a value can be a string, a number, etc.;*
- (e, r, e') *states that relation* r *links entity* e *to entity* e'*.*

In RDF and OWL, entities are called individuals, values are called literals, and the distinction between properties and relations corresponds to the distinction in OWL between datatype properties and object properties. URIs are used to identify entities, classes, properties, and relations.

Figure 1 shows two knowledge graphs, one on the left and another on the right. It falls into Scenario A as it uses completely different schemas and it only shares proper names of entities (square boxes). The objective is to find the best alignement between the two. This is the most complex example considered by Atencia *et al* [3] (Fig. 8, p.23). It involves cyclic dependencies that have to be taken into account to find the correct alignment.

It is easy to find that `o1:z1` aligns with `o2:i1` because they are the unique entities in each KG to have `"Dupont"` as a value. From this first alignement, one can find two additional alignments: property `o1:lastname` with `o2:name`, and class `o1:Person` with `o2:Inhabitant`. It is more difficult to find that `o1:z3` aligns with `o2:i3`. `o1:z3` has lastname `"Dubois"` but both `o2:i2` and `o2:i3` have name `"Dubois"`. `o1:z3` has a home/address in Paris but so do `o2:i2` and `o2:i3`. Traversing deeper in the graph, we find that `o1:z3`'s home is owned by a Dupont who lives in Grenoble. Something equivalent can be found for `o2:i3` but not for `o2:i2`. We observe that the alignment of entities, classes, properties

and relations depend on each other, and that handling long-range dependencies may be necessary to resolve alignments.

The example in Figure 1 can be adapted to cover Scenario B where string values are in different incomparable languages but some pre-aligned pairs may be available. Imagine for instance that the strings of the left-side KG are in Chinese, and that we already know that `o1:z1` aligns with `o2:i1`. Without that pre-aligned pair, one only has structural information, which is not enough to discriminate between the different people and places. Knowing that `o1:z1` aligns with `o2:i1` (Dupont), one can align classes `o1:Person` and `o2:Inhabitant`, and also relations `o1:home` and `o2:address`. By propagating alignments through neighborhood, starting from the pre-aligned pair, it is here possible to find the correct alignment for all elements: first `o1:h1` with `o2:a1`, then `o1:city` with `o2:city` and `o1:z2` with `o2:i2`, and so on.

3 Related Work

The alignment of KGs (or entities, or ontologies) is a matter of defining and combining similarity measures [2]. There are four kinds of similarities, depending on the kind of information that is used. Terminological similarity is based on the strings that are used to name entities and schema elements. Structural similarity is based on the description of entities by classes, properties, and relations. Extensional similarity is based on the extensions of classes. Semantic similarity is based on external sources of knowledge, and on automated reasoning (e.g., subsumption checks in description logics). Terminological and structural similarities are the most common methods, and are often combined to get better results. In general, strutural similarity does not consider the KG as a whole but use either tree-shaped descriptions extracted from the knowledge graph, or neighbor-based propagation of similarity values (e.g., RiMOM-IM [13]).

The trend in KG alignement is to use representation learning methods, in a supervised setting where a significant number of pre-aligned pairs is supposed to be known [14]. The principle is first to learn embeddings for the two KGs that coincide for the pre-aligned pairs. Other alignments can then be found by nearest-neighbor search in the embedding space. Many representation learning methods exist, such as TransE [4], ComplEx-N3 [10] or R-GCN [12].

Different variants of FCA have been applied to tasks related to KG alignment. Classical FCA was used for aligning (aka. matching) several biomedical ontologies, in a system called FCA-Map [15]. The scenario is here to match classes and relations based on their names and their relationships (e.g., class hierarchies, class disjointness, relation domains and ranges), there are no entities per se. The intuition behind using FCA is that if a concept has only two objects in its extent (here classes and relations), one from each ontology, then this is a strong indication that the two entities should be aligned. This is called an *anchor* in FCA-Map. The global alignment is obtained by defining a succession of five formal contexts, and computing their concept lattices. This can be seen

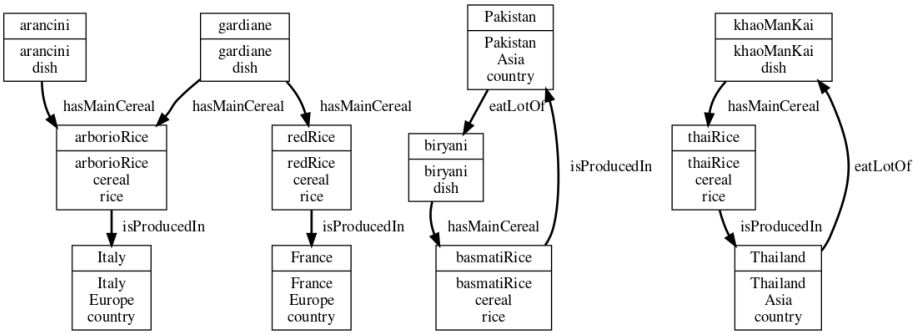


Fig. 2. Graph context about dishes, cereals, and countries.

as an ad-hoc form of Relational Concept Analysis (RCA) [11], with the notions of iterative concept formation, and relational attributes.

RCA was used for the task of *link key discovery* [3], which consists in finding pairs of properties and/or relations that are identifying keys for the entities of a pair of classes. In Figure 1, the pair `o1:lastname`/`o2:name` is not a link key for the pair of classes `o1:Person`/`o2:Inhabitant`, but combined with the pair `o1:home`/`o2:address`, it is. Link key discovery is somewhat stronger than KG alignment because in addition to provide alignments, it provides a rule for predicting aligned pairs in unseen data. RCA can be compared to methods that propagate similarity values through relations, except that formal concepts are used in place of numerical values. Pattern Structures [6] were used for the same task with an approach that is similar but does not cover long-range and cyclic dependencies like in RCA. In those approaches, the discovery of an alignment between two KGs is somewhat contrived. First, although a KG can be naturally represented as an RCA context, the RCA context must instead represent the product of the two KGs: i.e., the objects are all pairs of entities (one from each KG), the properties are all pairs of properties, and so on. This entails a quadratic increase in the size of the data, which poses scalibility issues. Second, the discovered alignment is scattered in the extents and intents of formal concepts, across several concept lattices. We show in this paper that those two disadvantages are overcome when using Graph-FCA [5] on the task of KG alignment.

4 Graph-FCA

Graph-FCA [5] is an extension of FCA for multi-relational data, and in particular for knowledge graphs. Graph-FCA defines a *graph context* as an incidence relation between *tuples of* objects and attributes. A *graph context* is a triple $K = (O, A, I)$, where O is a set of *objects*, A is a set of *attributes*, and $I \subseteq O^* \times A$ is an *incidence relation* between object tuples $\overline{o} = (o_1, \ldots, o_k) \in O^*$, for any arity k, and attributes $a \in A$. A graph context represents a KG with objects as nodes, and attributes as hyper-edge labels. An hyper-edge $((o_1, \ldots, o_k), a)$ can

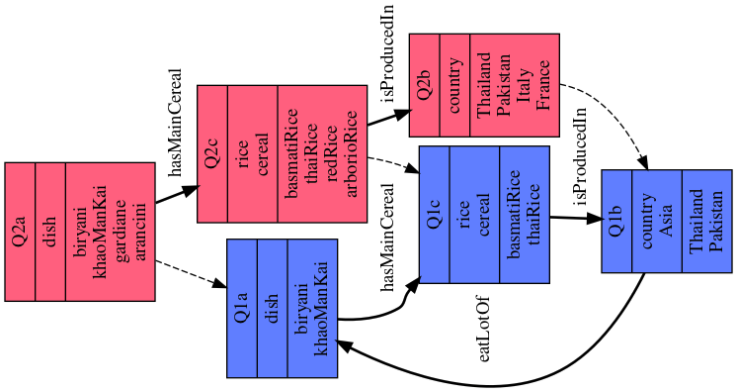


Fig. 3. Compact representation of graph concepts about dishes, cereals, and countries.

be seen as the logical atom $a(o_1, \ldots, o_k)$; we use the latter notation for readability. Unary edges correspond to node labels while binary edges correspond to labeled edges. Classical FCA is a special case of Graph-FCA where $k = 1$ (only unary edges) and $n = 1$ (only unary concepts).

Figure 2 shows the graphical representation of a graph context about dishes, cereals, and countries. Each box represents an object o along with the list of attributes a such that there is an unary edge $a(o) \in I$. Each arrow from box o_1 to box o_2 labelled by attribute a denotes a binary edge $a(o_1, o_2) \in I$. An hyperedge $a(o_1, \ldots, o_k)$ with $k > 2$ is represented as an ellipsis labeled by a, and having an arrow labeled i to each box o_i (not illustrated here).

Concept intents express what several n-tuples of objects have in common. They are graph patterns with n distinguished nodes. They are called *Projected Graph Patterns* (PGP), and they are similar to conjunctive queries. For example, the PGP $[(x, y) \leftarrow parent(x, z), parent(y, z)]$ has three nodes x, y, z, two edges $parent(x, z)$ and $parent(y, z)$, and two distinguished nodes x, y. It can be used as a definition of the "sibling" relationship, i.e. the fact that x and y are siblings if they have a common parent z. FCA is extended to graph contexts by extending the set operations to PGPs. PGP inclusion $\subseteq_q$ is based on graph homomorphisms, and PGP intersection $\cap_q$ is based on the *categorical product* of graphs (see [5] for details).

The Galois connection is defined in Graph-FCA between PGPs $(\mathcal{Q}, \subseteq_q)$ and sets of object tuples $(2^{O^*}, \subseteq)$. In the definitions of Q' and R' below, the PGP $[\overline{o} \leftarrow I]$ represents the description of an object tuple $\overline{o}$ by the whole incidence relation I seen from the relative position of $\overline{o}$.

$$Q' := \{\overline{o} \in O^n \mid Q \subseteq_q [\overline{o} \leftarrow I]\}, \text{ for } Q = [(x_1, \ldots, x_n) \leftarrow P] \in \mathcal{Q}$$
$$R' := \cap_q \{[\overline{o} \leftarrow I] \mid \overline{o} \in R\}, \qquad \text{for } R \subseteq O^n, n \in \mathbb{N}$$

Informally, Q' can be understood as the result set of Q, seen as a query; and R' as the most specific query whose result set contains R. From there, concepts

can be defined in the usual way, and organized into lattices. A concept is a pair (R, Q) such that $Q' = R$ and $R' =_q Q$. The arity of Q and R must be the same, it determines the arity of the concept. Unary concepts are about sets of objects, while binary concepts are about relationships between objects, and so on. Unlike RCA, there is a concept lattice for each concept arity rather than for each object type.

Figure 3 displays a compact representation of the graph concepts about dishes, cereals, and countries. It only shows unary concepts (R, Q), with $Q = [x \leftarrow P]$, s.t. $|R| \geq 2$, $P \neq \emptyset$, and P is a "core pattern" (i.e., P has no homomorphic subset). Each box x identifies a unary concept, with its name at the top (e.g., Q1b), and its extent at the bottom (here, *Pakistan*, *Thailand*). The concept intent is the PGP $[x \leftarrow P]$, where P is the subgraph made of solid arrows containing node x. Those arrows denote binary edges while the middle part of the boxes denote unary edges. By reading the graph, we learn that Concept Q1b is the concept of *"Asian countries, which eat a lot of some dish whose main cereal is a rice produced in the country itself"*. Formally, its intent is denoted by $[b \leftarrow country(b),\ Asia(b),\ eatLotOf(b, a),\ dish(a),\ hasMainCereal(a, c),\ cereal(c), rice(c), isProducedIn(c, b)]$ (using the small letters of concept names as pattern variables). Concepts Q1a and Q1c have the same graph pattern as Q1b but with a different focus, on dishes for Q1a and on cereals for Q2c. N-ary concepts are obtained by picking several distinguished nodes. For example, (Q1b,Q1a,Q1c) is a ternary concept whose instances are the object triples $(Pakistan, biryani, basmatiRice)$ and $(Thailand, khaoManKai, thaiRice)$. It represents the cyclic relationship existing between a country, a dish, and a cereal in Asian countries. The Q2-concepts are generalizations of the Q1-concepts. For example, concept Q2b covers all countries. The dashed arrow from Q2b to Q1b says that Q1b is a subconcept of Q2b. The Q2-intents say that, in the context, all dishes use some rice as a main cereal, and that all cereals are produced in some country but that not all countries eat a lot of some dish, and not all dishes are eaten a lot by some country.

5 Concept-based Knowledge Graph Alignment

In this section, we explain how KG alignments can be extracted from Graph-FCA concepts, and we discuss the results on the example and scenarios of Section 2.

5.1 Principle

When two entities from two different KGs are equivalent, e.g. `o1:z1` and `o2:i1` in the example, they generally have equivalent types (e.g., `o1:Person` and `o2:Inhabitant`), equivalent properties with equal values (e.g. `o1:lastname` and `o2:name` "Dupont"), and equivalent relations with equivalent entities (e.g., `o1:home` `o1:h1` and `o2:address` `o2:a1`). Similarly, when two classes are equivalent, they are the type of equivalent entities. In FCA terms, the two equivalent

entities (or classes) belong to the extent of a formal concept whose intent expresses everything they have in common. Here, the intent is not only about how the entities are described but also about how the entities are related to other entities. Moreover, the equivalence between two entities depend on the equivalence between the related entities, and so on recursively. This makes Graph-FCA, like RCA, a good match for the problem of KG alignment.

More precisely, let G_1 and G_2 be two KGs. If an unary concept $C = (R, Q)$ has only two elements in its extent, $R = \{x_1, x_2\}$, where x_1 belong to G_1 and x_2 belongs to G_2, then this is a strong indication that x_1 and x_2 are equivalent because they are unique in their own KG to match the projected graph pattern Q. Such a concept is called an *anchor* in [15]. Concepts that are not anchors can also contribute to the alignement, although in a less direct and less certain way. For example, a concept with extent $\{x_1, y_1, x_2, y_2\}$ says that G_1-entities x_1 and y_1 align with G_2-entities x_2 and y_2, without knowing which aligns to which. However, if we also have the anchor $\{x_1, x_2\}$, we can deduce an alignment between y_1 and y_2. Another example is a concept with extent $\{x_1, y_2, z_2\}$ where some ambiguity remains on the alignment of x_1 but it nonetheless reduces the set of possibilities from all G_2-entities to only two entities, y_2 and z_2.

5.2 Graph-FCA Representation

The immediate representation of a KG as a Graph-FCA context is to use classes, properties and relations as attributes (hyper-edge labels), and entities as objects (nodes). Each occurrence of a value is represented by a distinct node labelled by the value as this avoids spurious connections between objects. This is formalized by the following mappings from KG triples to Graph-FCA hyper-edges:

$$(e, \texttt{type}, c) \rightsquigarrow c(e) \qquad (e, p, v) \rightsquigarrow p(e, "v") \qquad (e, r, e') \rightsquigarrow r(e, e')$$

where $p(e, "v")$ is a shorthand for $p(e, o), v(o)$ for some fresh object o.

This immediate representation, however, does not produce any alignment because the two KG representations do not share any attribute. Indeed, the two KGs have different schemas, hence different classes, properties and relations. Recall that Graph-FCA concept intents are made of attributes and variables. As a consequence, for all concepts except the top one, the concept extent contains only objects from a single KG, and no anchor concept can be found.

An adequate representation should use the same set of attributes for the two KGs to be aligned, and those attributes should reflect what they have in common. In Scenario A, the two KGs only share the values. Actually, they also share the graph structure, and the distinction between classes, properties and relations. Anything else must hence be represented as objects: i.e., not only entities but also classes, properties and relations. We achieve this by introducing three meta-level common attributes: *type* for expressing the relation between entities and classes, *prop* for relating entities to values through properties, and *rel* for relating entities to entities through relations. The new mapping from KG triples to Graph-FCA hyper-edges is hence as follows:

$$(e, \texttt{type}, c) \rightsquigarrow \mathit{type}(e, c) \quad (e, p, v) \rightsquigarrow \mathit{prop}(e, p, "v") \quad (e, r, e') \rightsquigarrow \mathit{rel}(e, r, e')$$

This is a form of reification, in which classes, properties and relations are treated as objects. It is made possible by the fact that Graph-FCA supports hyper-edges beyond binary edges. As defined above, the notation "v" for a value implicitly introduces an attribute v, hence accounts for the values shared between KGs.

In Scenario B, in contrast to Scenario A, even the values are different in the two KGs. This means that values do not bring any information so that the mapping for properties can be simplified as $(e, p, v) \rightsquigarrow \mathit{prop}(e, p)$. Obviously, the two mappings can be mixed if some values are shared and others not. To compensate for the lack of shared elements, a set of pre-aligned pairs (x_1, x_2) may be given as input, and modelled as equivalence triples $(x_1, \texttt{sameAs}, x_2)$. Such an equivalence can be represented in Graph-FCA by using a fresh attribute a_x that represents the real entity or schema element x that x_1 and x_2 refer to. The mapping rule is

$$(x_1, \texttt{sameAs}, x_2) \rightsquigarrow a_x(x_1), a_x(x_2), \quad \text{for some fresh attribute } a_x.$$

Attribute a_x works like a nominal attribute for each KG, labelling only x_1 in G_1, and only x_2 in G_2.

5.3 Results and Discussion

We ran our Graph-FCA implementation[1] on the two scenarios in order to find anchors, and hence KG alignments. We used the following options to make their computation more efficient, and the output smaller: `-minsupp 2` because anchors have by definition a support equal to 2, and `-only-cores` because it excludes concepts that represent under-optimal alignments (an example is given below).

Scenario A (common values). Graph-FCA finds 24 unary concepts, grouped in only two patterns: Q1 and Q2. All anchor concepts are in Q1, and all concepts in Q1 are anchor concepts. Moreover, all elements of the two KGs (entities and schema) appear in those concepts. Pattern Q1 therefore represents a complete alignement between the two KGs. Table 1 lists those concepts with their identifier (a lowercase letter), their extent (an object from each KG), and their *neighboring intent*. For instance, concept g shows the alignment of `o1:z1` with `o2:i1`, and its intent reads as follows: The two entities have type l, property a with value "Dupont", relation d to entity h, and relation c from entity i. Each variable in the intent refers to a concept in the same pattern, and hence to an aligned pair: e.g., the common type l is `o1:Person`/`o2:Inhabitant`, and the property with value "Dupont" is `o1:lastname`/`o2:name`. One can observe that the alignement is also correct, not only for entities but also for classes, properties and relations.

The concepts in pattern Q2 are general concepts. One concept gathers all entities (people and places), another all classes, and similarly for properties and relations. It simply reflects the meta-model used in our representation. Without option `-only-cores`, Graph-FCA outputs additional concepts that represent approximate alignments, such as {`o1:z1, o1:z3, o2:i1, o2:i3`} or {`o1:lastname, o2:name, o2:given`}.

[1] Open source at https://bitbucket.org/sebferre/graph-fca/

Table 1. Anchor concepts in Scenario A (common values).

C	extent(C)		intent(C), only adjacent edges
a	o1:lastname	o2:name	$prop(e, a, "Dubois"), prop(f, a, "Dubois"), prop(g, a, "Dupont")$
b	o1:city	o2:city	$prop(h, b, "Grenoble"), prop(i, b, "Paris"), prop(j, b, "Paris")$
c	o1:owner	o2:ownedBy	$rel(h, c, f), rel(i, c, g), rel(j, c, e)$
d	o1:home	o2:address	$rel(e, d, i), rel(f, d, j), rel(g, d, h)$
e	o1:z3	o2:i3	$type(e, l), prop(e, a, "Dubois"), rel(e, d, i), rel(j, c, e)$
f	o1:z2	o2:i2	$type(f, l), prop(f, a, "Dubois"), rel(f, d, j), rel(h, c, f)$
g	o1:z1	o2:i1	$type(g, l), prop(g, a, "Dupont"), rel(g, d, h), rel(i, c, g)$
h	o1:h1	o2:a1	$type(h, k), prop(h, b, "Grenoble"), rel(g, d, h), rel(h, c, f)$
i	o1:h3	o2:a3	$type(i, k), prop(i, b, "Paris"), rel(e, d, i), rel(i, c, g)$
j	o1:h2	o2:a2	$type(j, k), prop(j, b, "Paris"), rel(f, d, j), rel(j, c, e)$
k	o1:House	o2:Place	$type(h, k), type(i, k), type(j, k)$
l	o1:Person	o2:Inhabitant	$type(e, l), type(f, l), type(g, l)$

Scenario B (pre-aligned pairs). Without common values nor any pre-aligned pairs, Graph-FCA only finds the meta-concepts, like in pattern Q2 in the above scenario. Indeed, there is no rationale for making any alignment. When adding a pre-aligned pair, like `o1:z1` with `o2:i1`, Graph-FCA finds an almost complete alignment (see Table 2). The only uncertainty is about `o1:lastname` that aligns with either `o2:name` or `o2:given`. Indeed, without values, the different properties that apply to an entity cannot be distinguished. Choosing any other pair of entities as pre-aligned pair gives the same result. However, if the pre-aligned pair is about the schema (e.g., two equivalent classes), then only the schema elements are correctly aligned. Entities are grouped by type (people and places) but they are not aligned individually.

Other results. We also applied our approach to the example of Abbas *et al* [1], which falls in Scenario A (common values). It is at the same time simpler than the example of Atencia *et al* because it has no relations (only classes and properties), and more difficult because there is not a one-to-one mapping between the two KGs. First, there are entities in G_1 that are not present in G_2. Second, the classes are not equivalent: the G_1-class `o1:Book` is split in two G_2-classes, `o2:Dictionary` and `o2:Novel`; and the G_2-class `o2:FemaleScientist` is equivalent to the intersection of two G_1-classes, `o1:Woman` and `o1:Scientist`. The alignments produced by Graph-FCA are grouped in two patterns, one for people and another for books, because there are no relations between them in the data. All common entities and all properties are completely and correctly aligned (one anchor concept for each pair). For classes, three interesting concepts are found: {`o1:Woman, o1:Scientist, o2:FemaleScientist`}, {`o1:Book, o2:Novel`}, and {`o1:Book, o2:Dictionary`}. They can be read as "Woman and Scientist together align with FemaleScientist", and "Book

Table 2. Anchor concepts in Scenario B (pre-aligned pairs).

C	extent(C)		intent(C), only adjacent edges
a	o1:city	o2:city	$prop(g,a), prop(h,a), prop(i,a)$
b	o1:lastname	o2:name	$prop(e,b), prop(f,b), prop(l,b)$
		o2:given	
c	o1:owner	o2:ownedBy	$rel(g,c,f), rel(h,c,l), rel(i,c,e)$
d	o1:home	o2:address	$rel(e,d,h), rel(f,d,i), rel(l,d,g)$
e	o1:z3	o2:i3	$type(e,k), prop(e,b), rel(e,d,h), rel(i,c,e)$
f	o1:z2	o2:i2	$type(f,k), prop(f,b), rel(f,d,i), rel(g,c,f)$
g	o1:h1	o2:a1	$type(g,j), prop(g,a), rel(g,c,f), rel(l,d,g)$
h	o1:h3	o2:a3	$type(h,j), prop(h,a), rel(e,d,h), rel(h,c,l)$
i	o1:h2	o2:a2	$type(i,j), prop(i,a), rel(f,d,i), rel(i,c,e)$
j	o1:House	o2:Place	$type(g,j), type(h,j), type(i,j)$
k	o1:Person	o2:Inhabitant	$type(e,k), type(f,k), type(l,k)$
l	o1:z1	o2:i1	$z1i1(l), type(l,k), prop(l,b), rel(h,c,l), rel(l,d,g)$

aligns with either Novel or Dictionary". This is exactly as expected. Note that each entity with type `o1:Book` is correctly assigned to one of `o2:Novel` and `o2:Dictionary` (two distinct concepts), based on its properties ("author" for novels, "year" for dictionaries).

Discussion. Compared to the use of classical FCA, RCA or Pattern Structures, our approach does not build a context as the "product" of the two KGs, i.e. with all pairs of entities as objects, and all pairs of classes/properties/relations as attributes. Instead, the Graph-FCA context is built as the "union" of the two KGs. This entails that the size of our context is the *sum* of the size of the two KGs, rather than the *product*. This clearly favors the scalability of our approach.

A key enabling feature of Graph-FCA is the support for k-ary relations as it enables to reify the binary properties and relations as objects, along with entities, and to use them as arguments of meta-level ternary predicates (e.g., *prop*). Note that this is akin to RDF triples where properties belong to the same space as individuals. As a consequence, the alignment of classes, properties and relations (the schema) can be interpreted in the Graph-FCA output exactly like for entities, through concept extents. This is in contrast with other FCA-based approaches, where a different interpretation method must be used for the schema because it appears in concept intents. Although it can be difficult to read Graph-FCA output in general, a KG alignment can be read from the unary concept extents, which are sets of objects, like in classical FCA.

6 Conclusion and Perspectives

Graph-FCA is FCA for knowledge graphs, and we have shown in this paper that by adequately representing two KGs in terms of what they have in common, we can extract aligned pairs of entities, classes, properties and relations from Graph-FCA concepts in a uniform way. The results in this paper are limited to a few

encouraging illustrative examples in different scenarios, and much remains to be done to apply our approach to real KGs. A first issue is scalability. A research track is to only generate anchors, i.e. concepts that result from the intersection of two objects, one from each KG. A second issue is to allow for approximate matching of values, especially texts, as they often slightly differ from one KG to another.

References

1. Abbas, N., David, J., Napoli, A.: Discovery of link keys in RDF data based on pattern structures: Preliminary steps. In: Int. Conf. Concept Lattices and Their Applications (2020)
2. Ardjani, F., Bouchiha, D., Malki, M.: Ontology-alignment techniques: Survey and analysis. Int. J. Modern Education & Computer Science **7**(11) (2015)
3. Atencia, M., David, J., Euzenat, J., Napoli, A., Vizzini, J.: Link key candidate extraction with relational concept analysis. Discrete applied mathematics **273**, 2–20 (2020)
4. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)
5. Ferré, S., Cellier, P.: Graph-FCA: An extension of formal concept analysis to knowledge graphs. Discrete Applied Mathematics **273**, 81–102 (2019)
6. Ganter, B., Kuznetsov, S.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) Int. Conf. Conceptual Structures. pp. 129–142. LNCS 2120, Springer (2001)
7. Ganter, B., Wille, R.: Formal Concept Analysis — Mathematical Foundations. Springer (1999)
8. Hahn, G., Tardif, C.: Graph homomorphisms: structure and symmetry. In: Graph symmetry, pp. 107–166. Springer (1997)
9. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
10. Lacroix, T., Usunier, N., Obozinski, G.: Canonical tensor decomposition for knowledge base completion. In: Dy, J.G., Krause, A. (eds.) Int. Conf. Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 2869–2878. PMLR (2018)
11. Rouane-Hacene, M., Huchard, M., Napoli, A., Valtchev, P.: Relational concept analysis: mining concept lattices from multi-relational data. Annals of Mathematics and Artificial Intelligence **67**(1), 81–108 (2013)
12. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: The Semantic Web Conf. (ESWC). pp. 593–607. Springer (2018)
13. Shao, C., Hu, L.M., Li, J.Z., Wang, Z.C., Chung, T., Xia, J.B.: Rimom-im: A novel iterative framework for instance matching. J. computer science and technology **31**(1), 185–197 (2016)
14. Zeng, K., Li, C., Hou, L., Li, J., Feng, L.: A comprehensive survey of entity alignment for knowledge graphs. AI Open **2**, 1–13 (2021)
15. Zhao, M., Zhang, S., Li, W., Chen, G.: Matching biomedical ontologies based on formal concept analysis. J. biomedical semantics **9**(1), 1–27 (2018)