A Proposal for Extending Formal Concept Analysis to Knowledge Graphs

Sébastien Ferré

A Proposal for Extending Formal Concept Analysis to Knowledge Graphs

Sébastien Ferré*

IRISA/Université de Rennes 1
Campus de Beaulieu, 35042 Rennes cedex, France
ferre@irisa.fr

Abstract. Knowledge graphs offer a versatile knowledge representation, and have been studied under different forms, such as conceptual graphs or Datalog databases. With the rise of the Semantic Web, more and more data are available as knowledge graphs. FCA has been successful for analyzing, mining, learning, and exploring tabular data, and our aim is to help transpose those results to graph-based data. Previous FCA approaches have already addressed relational data, hence graphs, but with various limits. We propose G-FCA as an extension of FCA where the formal context is a knowledge graph based on n-ary relationships. The main contributions is the introduction of "n-ary concepts", i.e. concepts whose extents are n-ary relations of objects. Their intents, "projected graph patterns", mix relationships of different arities, objects, and variables. In this paper, we lay first theoretical results, in particular the existence of a concept lattice for each concept arity, and the role of relational projections to connect those different lattices.

Keywords: Formal Concept Analysis, Knowledge Graph, Semantic Web, Graph Pattern, Relation, Projection

1 Introduction

Since the dawn of artificial intelligence, graphs have been used to represent knowledge as a set of interlinked entities. Notable formalisms are semantic networks, conceptual graphs [5], description logics [2], and more recently the Semantic Web [11]. In the last ten years, the number and size of knowledge graphs has exploded with the development of the Semantic Web, and its W3C standards (e.g., RDF, SPARQL). Its open side, called Linked Open Data (LOD), is now made of more than 1000 datasets [17], which contain about 70 billions semantic links (called triples). This effort has been joined by Web giants such as the Google Knowledge Graph or Facebook Graph Search.

Formal Concept Analysis (FCA) [10] is concerned with the definition of concepts from factual data, and their organization into a generalization ordering,

* This research is supported by ANR project IDFRAud (ANR-14-CE28-0012-02).

the concept lattice. It serves many purposes such as knowledge discovery, machine learning, information retrieval, or software refactoring. It seems therefore important to investigate the application of FCA to knowledge graphs. Only a few works consider its direct application to graphs. A power context family [18] is a form of knowledge graph, but there is a distinct concept lattice for each relation arity. Relational Concept Analysis (RCA) [16] defines concepts that mix unary and binary relationships, but only in tree-shaped patterns. Graphs have also been used as object descriptions (e.g., for molecules) as an application of pattern structures [9,13], but we do not consider those as knowledge graphs because graph nodes (e.g., individual atoms) are not formal objects. In all thoses approaches, only unary concepts are defined, i.e., extents are sets of objects, not relations. Concept lattices of relational structures [12] is an approach based on category theory that shares with us the use of n-ary relations, and relations as extents. However, the representation of an extent is not self-contained because it shares variables with the intent. Similarly, in RCA, an intent contains relational attributes that refer to other concepts, and so on, possibly in a circular way.

In this paper, we propose an extension of FCA, called Graph-FCA (G-FCA for short), that is directly applicable to knowledge graphs. Graph entities play the role of FCA objects, and graph relationships play the role of FCA attributes. The consequence is that the incidence relation relates tuples of objects (with various arities) to attributes, rather than single objects to attributes. A key novelty of our proposal is that an extensional representation is not a set of objects, but a set of *tuples of* objects, i.e. a n-ary relation. The particular case of unary relations corresponds to sets of objects. An intensional representation is defined as a *projected graph pattern* (PGP), i.e. as a graph pattern plus a *projection tuple*. The projection tuple can have any arity, and the graph pattern can mix relations with various arities. Both extensional and intensional representation are self-contained. A G-FCA concept is a pair (extent, intent) where the extent is an object relation, and the intent is a PGP. This significantly extends previous FCA approaches because power context families do not mix arities in concept definitions, and RCA only defines unary concepts based on unary and binary relations. In fact, PGPs are analogous to Datalog (non-recursive) predicate definitions [4], and to SPARQL queries. It suggests that G-FCA could be the basis for discovering or learning n-ary predicate definitions, and for querying knowledge graphs. The former is akin to Inductive Logic Programming (ILP) [15], and for the latter, we have already worked out a solution [6]. This paper aims at providing a formal basis and starting point for those applications.

After some technical preliminaries (Section 2), we formalize knowledge graphs as *graph contexts* (Section 3). We then introduce *projected graph patterns (PGP)* and *object relations*, and define mappings from one to the other based on *PGP inclusion* and *PGP intersection* (Section 4). From those definitions, we organize PGPs into a bounded lattice, and object relations into a complete lattice, from which we prove the existence of a concept lattice for each concept arity (Section 5). We relate the different concept lattices through projections (Section 6). Finally, we conclude and discuss future work on G-FCA (Section 7).

2 Tuples, Substitutions, and Projections

A *tuple* is noted $\overline{x} = (x_1, \ldots, x_k)$, where $|\overline{x}| = k$ is its *arity*. To avoid confusion with other kinds of indices, $\overline{x}[i]$ can be used as an alternate notation for x_i. The set of all k-tuples over a domain E is noted E^k. The set of all tuples is defined by $E^* = \bigcup_{k \geq 0} E^k$. There is only one 0-tuple, denoted by $()$. We use $1..k$ to denote the set of integers from 1 to k.

We assume an infinite set of variables $\mathcal{V}$, and we use letters to denote them (e.g., x, y). A *substitution* $\sigma \in \Sigma_E$ is a mapping from variables to elements of E. A substitution σ can be applied as a function to any structure, and returns that same structure with any variable x in it replaced by $\sigma(x)$. For example, given the substitution $\sigma = \{x \mapsto 1, y \mapsto z\}$, we have $\sigma((x, y, z)) = (1, z, z)$. The empty substitution is denoted by id, and the composition of two substitutions is denoted by $\sigma_2 \circ \sigma_1$, where $(\sigma_2 \circ \sigma_1)(x) = \sigma_2(\sigma_1(x))$. Given two k-tuples $\overline{x}, \overline{y}$, the notation $\sigma_{\overline{x}}^{\overline{y}}$ denotes the substitution that maps x_i to y_i, for every $i \in 1..k$, and any other variable not in $\overline{x}$ to itself. It is only well-defined when $\overline{y}$ is compatible with $\overline{x}$, i.e. for all $i \neq j \in 1..k$, $x_i = x_j \Rightarrow y_i = y_j$, and for all $i \in 1..k$, $x_i \notin \mathcal{V} \Rightarrow y_i = x_i$.

A *projection* $\pi \in \Pi_k^l$ is a function from target indices $1..l$ to source indices $1..k$. The projection $\pi_1 = \{1 \mapsto 3, 2 \mapsto 1\}$ can be more concisely represented by the tuple $(3, 1)$. Projections are used to map a tuple to another tuple according to the following formula: $\pi(\overline{x})[i] = \overline{x}[\pi(i)]$, i.e. the i-th element of a projected tuple is the element at index $\pi(i)$. The identity projection is denoted by $id_k \in \Pi_k^k$, and the composition of two projections is denoted by $\pi_2 \circ \pi_1$, where $(\pi_2 \circ \pi_1)(\overline{x}) = \pi_2(\pi_1(\overline{x}))$ (i.e., $(\pi_2 \circ \pi_1)(i) = \pi_1(\pi_2(i))$). A *permutation* is a bijective projection π, and has an inverse projection π^{-1} that is the inverse permutation. Note that the combined applications of a substitution σ and a projection π commute, i.e. $\pi(\sigma(\overline{x})) = \sigma(\pi(\overline{x}))$ for every substitution σ, projection $\pi \in \Pi_k^l$, and tuple $\overline{x} \in E^k$.

3 Knowledge Graphs as G-FCA Contexts

The first step is to formalize a knowledge graph as a formal context, which we call a *graph context*. The only difference with the classical FCA definition lies in the use of object tuples (O^*) instead of objects (O) in the incidence relation.

Definition 1 (graph context). *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 and attributes. The maximum cardinality of incidences is denoted by $|K|$.*

Figure 1 shows the graphical representation of a small graph context about USA presidents. It uses a notation similar to conceptual graphs, using rectangles for entities, and ellipses for relations [5]. A graph context is a directed multi-hypergraph, where each node is labelled by an object $o \in O$ (e.g., "Obama", "Hawaii", "2009"), and where each directed hyperedge is an incidence $(\overline{o}, a) \in I$.

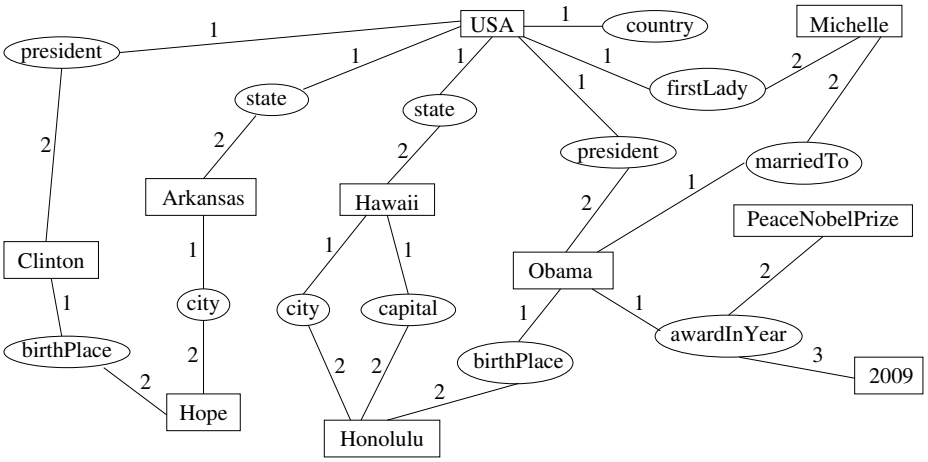

Fig. 1. Graphical representation of a graph context about USA presidents.

A hyperedge connects a number of objects $\overline{o}$ in a fixed order, and is labelled by an attribute $a \in A$ (e.g., "has president", "is a country"). If $|\overline{o}| = 2$, it is equivalent to a classical edge linking two nodes: e.g., $((\mathit{USA}, \mathit{Obama}), \mathit{president})$. If $|\overline{o}| = 1$, it is equivalent to a classical node labelling: e.g., $((\mathit{USA}), \mathit{country})$. The advantage of this definition is therefore to treat uniformly classical node labels and edge labels, and to support hyperedges, i.e. n-ary relationships: e.g., $((\mathit{Obama}, \mathit{PeaceNobelPrize}, \mathit{2009}), \mathit{awardInYear})$. Hyperedges are directed such that each position in the tuple of objects $\overline{o}$ corresponds to a particular role in the relationship: e.g., $((\mathit{USA}, \mathit{Hawaii}), \mathit{state})$ holds while $((\mathit{Hawaii}, \mathit{USA}), \mathit{state})$ does not. Nothing forbids to use the same attribute with different arities, but this amounts to have different relationships with the same name[1].

Graph contexts can easily be translated to/from other well-known relation-based representations. For example, a graph context is equivalent to the union of all elements of a power context family. The main limit when translating to other representations is for n-ary relationships with $n \geq 2$. That limit can be addressed by reifying hyperedges as nodes. In the Semantic Web, objects are RDF nodes (URIs, literals, and blank nodes), attributes are class and property URIs, and incidences are triples. In relational databases, objects are keys and values, attributes are table names, and incidences are table rows. In Inductive Logic Programming and Datalog, objects are Prolog atoms, attributes are predicates, and incidences are background knowledge facts. In RCA, objects are objects, attributes are either context attributes or relation names, and incidences are either $((o), a)$ when o has attribute a in some context, or $((o_1, o_2), r)$ when (o_1, o_2) are in relation r.

[1] Similarly to Prolog where predicates are identified by their name *and* arity.

4 Projected Graph Patterns and Object Relations

In this section, we introduce the intensional and extensional representations of G-FCA, as an extension of the sets of attributes and sets of objects of FCA. Section 4.1 defines *projected graph patterns* as an extension of the sets of attributes of FCA. Section 4.2 defines *object relations* as an extension of the sets of objects of FCA. Section 4.3 then defines a mapping from projected graph patterns to object relations, and Section 4.4 defines an inverse mapping from object relations to projected graph patterns. The two mappings are shown to form a Galois connection in Section 5.3, and are the basis of G-FCA concept lattices.

4.1 Projected Graph Patterns as Intensional Representations

A concept intent is an intensional representation that describes everything that a set of objects have in common. As a particular case, the intent of a single object is the description of that object. Depending on the FCA variant, an intensional representation can be a set of attributes, a logical formula [7], or a structure [9]. So, in order to identify G-FCA intensional representations, we may start by asking what is an adequate object description in G-FCA. An object (e.g., *USA*) should at least be described by its adjacent hyperedges (e.g., *having a president*) and adjacent objects (e.g., *Obama*). Then, if adjacent objects are interlinked (e.g., *USA's president* and *USA's first lady* are *married*), this should also appear in the description. Similarly, the descriptions of adjacent objects (e.g., *Obama being born in Honolulu*) should be included as they indirectly impact what the object is (e.g., *USA having a president born in Hawaii*).

All in all, this implies that the description of an object is the entire knowledge graph, or at least the connected component it belongs to if the knowledge graph is disconnected. Given that knowledge graphs, like the Web, are generally not disconnected, this seems to imply that all objects have the same description! In fact, it is like if all objects were represented by the same knowledge graph, the same world, but each from a different point of view. The different points of view can be interpreted as different phrasings of the same information: e.g., "Obama is the president of USA, and was born in Honolulu", and "Honolulu is the birth place of the president of USA, Obama". Therefore, the description of an object o in a graph context $K = (O, A, I)$ can be defined as the couple (o, I).

It is possible to generalize descriptions from objects to tuples of objects. For example, the description of the 2-tuple (*Honolulu*, *USA*) should contain the properties of each object, and the relationships (direct and indirect) that link them (e.g., Honolulu "is the capital of a state of" USA, Honolulu "is the birth place of a president of" USA). Similarly to single objects, the description of a tuple of objects $\overline{o}$ can be defined as $(\overline{o}, I)$.

Object descriptions need to be generalized to form concept intents shared by several objects, or several tuples of objects. For instance, we want to describe what *Obama* and *Clinton* have in common, or what (*Honolulu*, *Hawaii*) and (*Houston*, *Texas*) have in common. Like in ILP, generalization is obtained in two ways: (1) replacing objects by variables, and (2) removing hyperedges.

This leads us to the following definitions of a *graph pattern* as a generalized incidence relation, and of a *projected graph pattern* (PGP) as a generalized description. In the following definitions, we use $\mathcal{X} = O \cup \mathcal{V}$ to denote the domain of nodes in graph patterns, which can be objects or variables. We assume a graph context $K = (O, A, I)$ that defines objects and attributes.

Definition 2 (graph pattern). *A* graph pattern $P \subseteq \mathcal{X}^* \times A$ *is a set of directed hyperedges with variables and/or objects as nodes, and attributes as labels. Substitutions are extended to patterns:* $\sigma(P) = \{(\sigma(\overline{v}), a) \mid (\overline{v}, a) \in P\}$.

Graph patterns have the same type as incidence relations, only allowing variables in addition to objects as nodes. For instance, the pattern $P_{ex} = \{((x, y), \mathit{president}), ((y, \mathit{Honolulu}), \mathit{birthPlace})\}$ describes any situation where "some entity x has as a president another entity y, which has birth place *Honolulu*". Every graph pattern can be seen as a small graph context, abstracted over some objects by variables. We are primarily interested in connected patterns, but disconnected patterns are not excluded.

Definition 3 (projected graph pattern). *A* projected graph pattern *(PGP) is a couple* $Q = (\overline{x}, P)$ *where* P *is a graph pattern, and* $\overline{x} \in \mathcal{X}^*$*, called* projection tuple*, is a tuple of variables and possibly objects.* $|Q| = |\overline{x}|$ *denotes the* arity *of the PGP. We note* $\mathcal{Q}$ *the set of PGPs, and* $\mathcal{Q}_k$ *the subset of PGPs having arity* k*. Projections are extended to PGPs:* $\pi(Q) = (\pi(\overline{x}), P)$.

A projection tuple can be seen as a tuple of objects abstracted with variables. For instance, the PGP $Q_{ex} = ((x), P_{ex})$ using the above pattern describes "any entity x having a president born in Honolulu". Pattern hyperedges can be seen as constraints on variables. A variable that occurs in the projection tuple but not in the pattern is unconstrained, and can take any object as value. A variable that occurs in the pattern but not in the projection tuple is existentially quantified with respect to projected variables. In Q_{ex}, there must exist an entity y that the president of x and is born in Honolulu, but which one and how many does not matter.

Objects and duplicates in the projection tuple $\overline{x}$ define equality constraints between the indices of $\overline{x}$.

Definition 4 (equality constraints). *Let* $\overline{x}$ *be a projection tuple of arity* k*. Its set of* equality constraints *is defined by*

$$Eq(\overline{x}) = \{(i, j) \mid i < j \in 1..k, x_i = x_j\} \cup \{(i, o) \mid i \in 1..k, o \in O, x_i = o\}.$$

A set of equality constraints generates an equivalence relation between projection tuple indices and objects, and is confused with it in the following.

For example, the set of equality constraints of the projection tuple (x, o, x, y) is $\{(1, 3), (2, o)\}$, and generates three equivalence classes: $\{1, 3\}$, $\{2, o\}$, and $\{4\}$. Equality constraints come in addition to hyperedge constraints from the graph

pattern P. By allowing objects in projection tuples, we allow object descriptions in the form $(\overline{o}, I)$ to act as fully instantiated PGPs. By allowing duplicates, we allow a single entity to play different roles, like when searching a common PGP between the pairs $(Canberra, Sydney)$ and $(Paris, Paris)$: e.g. $((x, y), \{((z), Country), ((z, x), capital), ((z, y), biggestCity)\})$.

As said above, PGPs are an extension of the sets of attributes of FCA. For instance, the set of attributes $\{a, c, d\}$ corresponds to the PGP $((x), \{((x), a), ((x), c), ((x), d)\})$, where x is a variable. Indeed, in FCA, $\{a, c, d\}$ covers all objects that have at least attributes a, c, d. In G-FCA, this can be expressed as a node labelled by a, c, d, hence the introduction of the variable x. In general, a set of attributes Y corresponds to the PGP $((x), \{((x), a) \mid a \in Y\})$.

PGPs are comparable to non-recursive predicate definitions, and to SPARQL ASK/SELECT conjunctive queries. For example, the definition of the 'uncle' predicate

$$uncle(x, y) :\Leftrightarrow \exists z.parent(x, z) \wedge brother(z, y)$$

is equivalent to the PGP $((x, y), \{((x, z), parent), ((z, y), brother)\})$. Similarly, the SPARQL SELECT query

```
SELECT ?x ?y WHERE
{ ?x a :Film . ?x :genre :ScienceFiction . ?x :director ?y }
```

is equivalent to the PGP $((x, y), \{((x), Film), ((x, ScienceFiction), genre), ((x, y), director)\})$. The SPARQL ASK queries (yes/no questions) correspond to PGPs whose arity is zero ($\overline{x} = ()$).

4.2 Object Relations as Extensional Representations

In our introduction of PGPs as intensional representations, we made a shift from single objects to tuples of objects. That implies that extensional representations are sets of tuples of objects. With the constraint that all member tuples have the same arity, we obtain that extensional representations are *object relations*.

Definition 5 (object relation). *An* object relation *is a set $R \subseteq O^k$, for some arity $|R| = k$, of object tuples. We note $\mathcal{R}$ the set of object relations, and $\mathcal{R}_k$ the subset of relations having arity k. Projections are extended to relations: $\pi(R) = \{\pi(\overline{o}) \mid \overline{o} \in R\}$.*

$\mathcal{R}_0$ has only two relations: $\{()\}$ and $\{\}$. It can be seen as the Boolean type, with the two relations meaning "true" and "false" respectively. $\mathcal{R}_1$ has one relation for each set of objects $X \subseteq O$. It therefore corresponds to FCA extensional representations. For instance, the set of objects $\{o_1, o_3, o_4\}$ corresponds to the object relation $\{(o_1), (o_3), (o_4)\}$, simply embedding each object into a 1-tuple. In general, a set of objects X corresponds to the object relation $\{(o) \mid o \in X\}$. Object relations are comparable to the interpretations of a predicate in classical logic, and to SPARQL query results.

4.3 From Patterns to Relations

We here define a mapping from PGPs to object relations, i.e. from intensional representations to extensional representations. It defines for each PGP its *extension*, i.e. its set of instances in a given graph context. An instance is a tuple of objects whose description $(\overline{o}, I)$ "contains" the PGP modulo a substitution.

Definition 6 (PGP inclusion). *Let $Q_1 = (\overline{x}_1, P_1)$, $Q_2 = (\overline{x}_2, P_2)$ be two PGPs with same arity: $|Q_1| = |Q_2|$. Q_1 is* included in *Q_2, or equivalently Q_2* contains *Q_1, which is denoted by $Q_1 \subseteq_q Q_2$ iff there exists an substitution σ s.t. $\sigma(\overline{x}_1) = \overline{x}_2$ and $\sigma(P_1) \subseteq P_2$.*

$$Q_1 \subseteq_q Q_2 :\Leftrightarrow \exists \sigma \in \Sigma_{\mathcal{X}} : \sigma(\overline{x}_1) = \overline{x}_2 \wedge \sigma(P_1) \subseteq P_2$$

That definition is careful to account for variable renamings by introducing a substitution from Q_1-variables to Q_2-nodes. Indeed, like in logical formulas, variable names are irrelevant to the meaning of PGPs.

Definition 7 (extension). *Let $K = (O, A, I)$ be a graph context. The* extension *of a k-PGP $Q = (\overline{x}, P)$, denoted by $ext(Q)$, is defined by*

$$ext(Q) := \{\overline{o} \in O^k \mid Q \subseteq_q (\overline{o}, I)\}$$

The above definitions say that for every occurence of the pattern P in the graph context ($\sigma(P) \subseteq I$), there is an instance of Q ($\sigma(\overline{x})$). Conversely, if $\overline{o}$ is an instance of Q, then $Q = (\overline{x}, P)$ must be a generalization of its description $(\overline{o}, I)$, i.e. replacing some objects by variables and relaxing some constraints.

For example, in the graph context of Figure 1, the PGP $((x, y), \{((\mathit{USA}, x), \mathit{president}), ((x, z), \mathit{birthPlace}), ((\mathit{USA}, y), \mathit{state}), ((y, z), \mathit{city})\})$ has the following extension: $\{(\mathit{Obama}, \mathit{Hawaii}), (\mathit{Clinton}, \mathit{Arkansas})\}$. That PGP retrieves the list of USA presidents along with the state of their birth place.

The above definition is compatible with the interpretation of a predicate definition in classical logic: our incidence relation I corresponds to a model, and our substitution σ corresponds to a variable assignment. It is also compatible with SPARQL query results: our incidence relation I corresponds to a RDF graph, and our substitution corresponds to a solution mapping. Finally, it is consistent with classical FCA in the case where only 1-tuples are used, i.e., when $Q = ((x), \{((x), a) \mid a \in Y\})$ for some set of attributes $Y \subseteq A$. Indeed, by casting 1-tuples to their element, and picking $\sigma = \{x \mapsto o\}$, we obtain the classical FCA definition: $ext(Y) = \{o \in O \mid \forall a \in Y : (o, a) \in I\}$.

Note that substitutions used in PGP inclusion are homomorphisms, and not isomorphisms, because two different variables can be substituted by a single node. This departs from previous work in graph mining and FCA [19,13] which are based on isomorphisms, but this follows classical logic and SPARQL querying as explained above.

4.4 From Relations to Patterns

We here define a mapping from object relations to PGPs, i.e. from extensional representations to intensional representations. It defines for each object relation its *intension* as the "PGP intersection" of the description of all tuples $\overline{o} \in R$.

Definition 8 (PGP intersection). *Let $\{Q_i = (\overline{x}_i, P_i)\}_{i \in 1..n}$ be a finite and non-empty collection of n PGPs of same arity k. Let ν be a fixed bijection from $\mathcal{X}^n$ to $\mathcal{X}$ s.t. $\nu(x, \ldots, x) = x$ for all $x \in \mathcal{X}$. The* PGP intersection $\cap_q\{Q_i\}_{i \in 1..n}$ *is the PGP $Q = (\overline{x}, P)$ of arity k, where:*

- $\overline{x} = (x_1, \ldots, x_k)$, *where for all* $j \in 1..k$, $x_j = \nu(\overline{x}_1[j], \ldots, \overline{x}_n[j])$,
- $P = \{((\nu(\overline{v}_1), \ldots, \nu(\overline{v}_k)), a) \mid k \in 1..|K|, \forall j \in 1..k : \overline{v}_j \in \mathcal{X}^n, a \in A,$
  $\forall i \in 1..n : ((\overline{v}_1[i], \ldots, \overline{v}_k[i]), a) \in P_i\}$.

Definition 9 (intension). *Let $K = (O, A, I)$ be a graph context. The* intension *of a non-empty object relation $R \in \mathcal{R}_k$, denoted by $int(R)$, is defined by*

$$int(R) = \cap_q\{(\overline{o}, I)\}_{\overline{o} \in R}$$

The idea of PGP intersection is to define a node $\nu(\overline{v}_j)$ for every possible alignment of n-nodes $\{\overline{v}_j[i]\}_{i \in 1..n}$, one from each PGP Q_i of the collection. Then, if an hyperedge holds at every position $i \in 1..n$ of k alignment nodes $(\overline{v}_1, \ldots, \overline{v}_k)$, then it is a shared structure and it belongs to the pattern of the PGP intersection. The projection tuple is then derived from the alignment of the projection tuples of the collection. In practice, the connected components of the graph pattern P that do not contain any element of the projection tuple can be omitted (reduced intension) because they do not affect the extension of the PGP intersection. For the same reason, hyperedges that are in the incidence relation I can be omitted.

For example, in the graph context of Figure 1, the object relation $R = \{(\mathit{USA}, \mathit{Hope}), (\mathit{USA}, \mathit{Honolulu})\}$ has the following reduced intension: $((\mathit{USA}, x), \{((\mathit{USA}, y), \mathit{state}), ((y, x), \mathit{city}), ((\mathit{USA}, z), \mathit{president}), ((z, x), \mathit{birthPlace})\})$.
The obtained variables are derived from the following alignments: $x = \nu(\mathit{Hope}, \mathit{Honolulu})$, $y = \nu(\mathit{Arkansas}, \mathit{Hawaii})$, $z = \nu(\mathit{Clinton}, \mathit{Obama})$. Through alignments, PGP intersection does not only provide a common PGP, but also an explanation of how each instance relates to the others. PGP intersection also applies to PGPs with variables. For example, the intersection of $Q_1 = ((x_1), \{((x_1, y_1), a), ((x_1, z_1), c), ((y_1, z_1), b)\})$ and $Q_2 = ((x_2), \{((x_2, y_2), a), ((x_2, y_2), c), ((y_2, y_2), b)\}$ is $Q = ((x), \{((x, y), a), ((x, z), c), ((y, z), b)\})$, which is isomorphic to Q_1. Here, both y_1, z_1 are aligned with y_2, hence generating two variables $y = \nu(y_1, y_2)$ and $z = \nu(z_1, y_2)$. Note that Q is not subgraph isomorphic to Q_2. Indeed, under isomorphism, PGP intersection would be the problem of Maximum Common Subgraphs (MCS). A drawback of MCS is that there is generally not a unique solution, so that sets of graph patterns have to be used for intensional representations [13]. Moreover, the MCSs can be less specific. For example, the MCSs of Q_1 and Q_2 patterns are $\{((x, y), a)\}$ and $\{((x, z), c)\}$.

The above definitions of PGP intersection and intension are consistent with classical FCA, where only unary relations (sets) are used. In this case, every PGP needs only one variable. Therefore, given a set of objects $X = \{o_i\}_{i \in 1..n} \subseteq O$, the intension of X has one projected variable $x = \nu(o_1, \ldots, o_n)$, and a graph pattern like $P = \{((x), a) \mid a \in A, \forall i \in 1..n : ((o_i), a) \in I\}$. By casting 1-tuples to their element, we obtain the classical FCA definition: $int(X) = \{a \in A \mid \forall o \in X : (o, a) \in I\}$.

5 A Family of Graph Concept Lattices

In order to prove that (ext, int) forms a Galois connection, and hence the existence of a *graph concept lattice* for each concept arity, we first define partial orderings for each arity, over both PGPs and object relations. We also show that PGPs form a bounded lattice, and object relations a complete lattice.

5.1 Lattices of k-PGPs

The partial ordering over PGPs should correspond to a generalization ordering over them. Intuitively, a PGP Q_1 is more general than a PGP Q_2 if Q_1 is included in Q_2: $Q_1 \subseteq_q Q_2$ (see Definition 6). Indeed, assume $Q_2 = ((x, y), \{((x), \mathit{country}), ((x, y), \mathit{president})\})$ representing the relationship between countries and their president. Then, $Q_1 = ((x, y), \{((x, y), \mathit{president})\})$ representing the relationship between different kinds of organizations and their president is more general than Q_2 because it relaxes the constraint saying that the organization should be a country. Generalization by constraint relaxation is also found in ILP to define subsumption between learning hypotheses.

Recall that PGP inclusion is defined modulo a substitution, and that a substitution can map two different nodes in Q_1 to a single node in Q_2. The latter corresponds to adding an equality constraint between two entities, which is indeed a specialization. It enables to have

$$((x, y), \{((x, y), \mathit{president}), ((x', y), \mathit{president})\}) \subseteq_q ((x, y), \{((x, y), \mathit{president})\}),$$

by mapping both x, x' to x. Note that the first PGP does not state that y is the president of two organizations, but rather states twice that y is a president, which is equivalent to the second PGP. As the reverse inclusion trivially holds, the two PGPs are equivalent representations of the same thing. We note $Q_1 \equiv_q Q_2$ when $Q_1 \subseteq_q Q_2$ and $Q_2 \subseteq_q Q_1$. PGP inclusion is compatible with inclusion between sets of attributes in FCA. Indeed, as a single variable is involved in FCA,the substitution must be the identity function, and the definition of $Q_1 \subseteq_q Q_2$ boils down to $P_1 \subseteq P_2$.

We prove that $\subseteq_q$ is a preorder, and hence that a partially ordered set is obtained for patterns by considering equivalence classes of PGPs modulo $\equiv_q$.

Lemma 1. *PGP inclusion $\subseteq_q$ is a preorder over PGPs.*

Proof. **reflexivity.** Given a PGP Q, it suffices to take $\sigma = id$ to verify $\sigma(P) \subseteq P$ and $\sigma(\overline{x}) = \overline{x}$, and hence $Q \subseteq_q Q$.

transitivity. Assume PGPs Q_1, Q_2, Q_3 s.t. $Q_1 \subseteq_q Q_2$ and $Q_2 \subseteq_q Q_3$. Hence, there exists two substitutions σ_1, σ_2 s.t. $\sigma_1(P_1) \subseteq P_2$, $\sigma_2(P_2) \subseteq P_3$, $\sigma_1(\overline{x}_1) = \overline{x}_2$, and $\sigma_2(\overline{x}_2) = \overline{x}_3$. Then, it suffices to take $\sigma = \sigma_2 \circ \sigma_1$ to verify $\sigma(P_1) = \sigma_2(\sigma_1(P_1)) \subseteq \sigma_2(P_2) \subseteq P_3$, and also $\sigma(\overline{x}_1) = \sigma_2(\sigma_1(\overline{x}_1)) = \sigma_2(\overline{x}_2) = \overline{x}_3$. Hence $Q_1 \subseteq_q Q_3$. ■

Before showing that the pre-ordering over k-PGPs forms a bounded lattice modulo $\equiv_q$, for every arity k, we first need to define *PGP union* to act as a supremum. To this purpose, we need to introduce an additional *maximal PGP*, denoted by Ω_q, that is defined as containing all PGPs ($\forall Q : Q \subseteq_q \Omega_q$), and only included in itself ($\forall Q : \Omega_q \subseteq_q Q \Rightarrow Q = \Omega_q$). It can therefore be used to extend the definition of PGP intersection to empty collections: $\cap_q \emptyset := \Omega_q$.

Definition 10 (PGP union). *Let $\{Q_i = (\overline{x}_i, P_i)\}_{i \in 1..n}$ be a finite collection of n PGPs of same arity k, using disjoint sets of variables. The* PGP union $\cup_q \{Q_i\}_{i \in 1..n}$ *is either the PGP $(\overline{x}, P)$ of arity k verifying*

- $Eq(\overline{x}) = \bigcup_{i \in 1..n} Eq(\overline{x}_i)$
- $P = \bigcup_{i \in 1..n} \{\sigma^{\overline{x}}_{\overline{x}_i}(P_i)\}$

when $Eq(\overline{x})$ has no two different objects in a same equivalence class; or else Ω_q.

The assumption that PGPs do not share any variable is there to avoid variable capture. It entails no loss of generality because variable can be renamed freely. PGP union corresponds to add both equality and edge constraints of all PGPs Q_i, and is logically equivalent to a conjunction. When a projection tuple that satisfies all equality constraints can be formed, it is used as a projection tuple of the PGP union, and also to merge variables ($\sigma^{\overline{x}}_{\overline{x}_i}$) from the different projection tuples $\overline{x}_i$ in the collection of PGPs. Given a set of equality constraints, e.g. $\{(1,3),(2,o)\}$, a projection tuple is formed by using a single node for all indices of an equivalence class (e.g., 1 and 3), and by choosing as a node the object in the equivalence class if there is one (e.g., o for 2), or a fresh variable otherwise (e.g., x for 1 and 3). The case where several objects belong to a same equivalence class corresponds to a contradiction between the differents PGPs, and the maximal PGP Ω_q is used to denote such a contradiction. The extension of Ω_q is always the empty relation because it is only included in itself, and not in any object tuple description. The PGP union of an empty collection corresponds to an empty set of constraints, and defines the *minimal PGP* $\emptyset_q = (\overline{x}, \emptyset)$, where $\overline{x}$ is a tuple of k distinct variables. Finally, PGP union is compatible with the union of sets of attributes in FCA.

We first prove two lemmas stating that $\cup_q$ and $\cap_q$ are respectively the supremum and infimum of k-PGPs, before stating the main theorem about bounded lattices of k-PGPs.

Lemma 2. *Let Q_1, Q_2 be two PGPs. Their PGP union $Q_1 \cup_q Q_2$ is their* supremum *relative to query inclusion $\subseteq_q$.*

Proof. Let $Q = Q_1 \cup_q Q_2$. To prove that Q is an upper bound, it suffices to prove that it contains both Q_1 and Q_2. If $Q = \Omega_q$, then it contains both Q_1 and Q_2 by definition. Otherwise $Q = (\overline{x}, P)$. To prove $Q_1 \subseteq_q Q$, it suffices to choose $\sigma_1 = \sigma^{\overline{x}}_{\overline{x}_1}$, which is well-defined because $Eq(\overline{x}_1) \subseteq Eq(\overline{x})$, and to prove $\sigma_1(\overline{x}_1) = \overline{x}$ and $\sigma_1(P_1) \subseteq P$. This is easily obtained from the definition of Q. The proof of $Q_2 \subseteq_q Q$ is identical with $\sigma_2 = \sigma^{\overline{x}}_{\overline{x}_2}$.

To prove that Q is the *least* upper bound (the supremum), we have to prove that every PGP Q' that contains both Q_1 (via σ_1) and Q_2 (via σ_2) also contains Q. If $Q' = \Omega_q$, then $Q \subseteq_q Q'$ by definition of Ω_q. Otherwise, $Q' = (\overline{x}', P')$. From hypotheses $\sigma_1(\overline{x}_1) = \overline{x}'$ and $\sigma_2(\overline{x}_2) = \overline{x}'$, we obtain that $Eq(\overline{x}_1) \subseteq Eq(\overline{x}')$ and $Eq(\overline{x}_2) \subseteq Eq(\overline{x}')$. Then, we have $Eq(\overline{x}) = Eq(\overline{x}_1) \cup Eq(\overline{x}_2) \subseteq Eq(\overline{x}')$, and hence that $\sigma = \sigma^{\overline{x}'}_{\overline{x}}$ is well-defined. We can then easily prove that $\sigma(\overline{x}) = \overline{x}'$ and $\sigma(P) \subseteq P'$, and hence that $Q \subseteq_q Q'$. ■

Lemma 3. *Let Q_1, Q_2 be two PGPs. Their PGP intersection $Q_1 \cap_q Q_2$ is their* infimum *relative to query inclusion $\subseteq_q$.*

Proof. To prove that $Q_1 \cap_q Q_2$ is a lower bound, it suffices to prove that Q is included in both Q_1 and Q_2. To prove $Q \subseteq_q Q_1$, it suffices to choose the substitution $\sigma_1(x) = (\nu^{-1}(x))[1]$, and to prove that $\sigma_1(\overline{x}) = \overline{x}_1$ and $\sigma_1(P) \subseteq P_1$. This is easily obtained from the definition of Q. The proof of $Q \subseteq_q Q_2$ is identical with $\sigma_2(x) = (\nu^{-1}(x))[2]$.

To prove that $Q_1 \cap_q Q_2$ is the *greatest* lower bound (the infimum), we have to prove that every PGP Q' that is included in both Q_1 (via σ_1) and Q_2 (via σ_2) is also included in Q. To that purpose, it suffices to choose $\sigma(x') = \nu(\sigma_1(x'), \sigma_2(x'))$, and to prove that $\sigma(\overline{x}') = \overline{x}$ and $\sigma(P') \subseteq P$. This can be obtained from the definition of Q, and from the hypotheses $\sigma_1(\overline{x}') = \overline{x}_1$, $\sigma_1(P') \subseteq P_1$, $\sigma_2(\overline{x}') = \overline{x}_2$, and $\sigma_2(P') \subseteq P_2$. ■

Theorem 1. *For every arity k, the algebraic structure $(\mathcal{Q}_k, \subseteq_q, \cap_q, \cup_q, \Omega_q, \emptyset_q)$ forms a bounded lattice, module $\equiv_q$.*

Proof. The proof follows immediately from above lemmas and definitions. ■

5.2 Complete Lattices of Object k-Relations

The partial ordering over object relations should be consistent with the partial ordering on PGPs if we are to obtain concept lattices. Therefore, it should correspond to a form of generalization at the extensional level. A PGP can be made more general by relaxing constraints, which entails a larger extension. As object relations are sets of object tuples, we simply use set inclusion to partially order them. Given that $\mathcal{R}_k$ is the powerset of O^k, the poset $(\mathcal{R}_k, \subseteq, \cap, \cup, O^k, \emptyset)$ is a complete lattice, with set intersection $\cap$ as infimum, set union $\cup$ as supremum, full relation O^k as top, and empty relation $\emptyset$ as bottom.

5.3 Lattices of Graph k-Concepts

In order to prove the existence of a concept lattice for each arity, it suffices to prove that the two mappings between extensional and intensional representations form a Galois connection.

Theorem 2 (Galois connection). *Let $K = (O, A, I)$ be a graph context. For every arity k, the pair of mappings (ext, int) forms a Galois connection between $(\mathcal{R}_k, \subseteq)$ and $(\mathcal{Q}_k, \subseteq_q)$, i.e. for every object relation $R \in \mathcal{R}_k$ and PGP $Q \in \mathcal{Q}_k$,*

$$R \subseteq ext(Q) \iff Q \subseteq_q int(R)$$

Proof. $R \subseteq ext(Q) \iff \forall \overline{o} \in R : \overline{o} \in ext(Q)$
$\iff \forall \overline{o} \in R : Q \subseteq_q (\overline{o}, I)$ (Definition 7)
$\iff Q \subseteq_q \cap_q \{(\overline{o}, I)\}_{\overline{o} \in R}$ (Lemma 3)
$\iff Q \subseteq_q int(R)$ (Definition 9) ■

Corollary 1. *From (ext, int) being a Galois connection and from $(\mathcal{R}_k, \subseteq, \cap, \cup, O^k, \emptyset)$ and $(\mathcal{Q}_k, \subseteq_q, \cap_q, \cup_q, \Omega_q, \emptyset_q)$ being lattices, we have the following propositions for every relations $R, R_1, R_2 \in \mathcal{R}_k$, and every PGP $Q, Q_1, Q_2 \in \mathcal{Q}_k$, for any arity k:*

(1a) $Q_1 \subseteq_q Q_2 \Rightarrow ext(Q_1) \supseteq ext(Q_2)$	(1b) $R_1 \subseteq R_2 \Rightarrow int(R_1) \supseteq_q int(R_2)$
(2a) $Q \subseteq_q int(ext(Q))$	(2b) $R \subseteq ext(int(R))$
(3a) $int(R) \equiv_q int(ext(int(R)))$	(3b) $ext(Q) = ext(int(ext(Q)$
(4a) $int(R_1 \cup R_2) \equiv_q int(R_1) \cap_q int(R_2)$	(4b) $ext(Q_1 \cup_q Q_2) = ext(Q_1) \cap ext(Q_2)$
(5a) $int(\emptyset) \equiv_q \Omega_q$	(5b) $ext(\emptyset_q) = O^k$

From the Galois connection, *graph concepts* can be defined and organized into concept lattices, like in classical FCA, with one concept lattice for each arity k.

Definition 11 (graph concept). *Let $K = (O, A, I)$ be a graph context. A* graph concept *of K is a pair (R, Q), made of an object relation (the* extent*) and a PGP (the* intent*), such that $R = ext(Q)$ and $Q \equiv_q int(R)$. The* arity *of a graph concept is the arity of its extent and intent, which have to be equal.*

Theorem 3 (graph concept lattices). *The set of graph k-concepts $\mathcal{C}_k$, partially ordered by $\leq$, which is defined by $(R_1, Q_1) \leq (R_2, Q_2) : \iff R_1 \subseteq R_2 \iff Q_2 \subseteq_q Q_1$, forms a bounded lattice $(\mathcal{C}_k, \leq, \wedge, \vee, \top, \bot)$ where:*

1. $(R_1, Q_1) \wedge (R_2, Q_2) = (R_1 \cap R_2, int(ext(Q_1 \cup_q Q_2))$,
2. $(R_1, Q_1) \vee (R_2, Q_2) = (ext(int(R_1 \cup R_2)), Q_1 \cap_q Q_2)$,
3. $\top = (O^k, int(ext(\emptyset_q)))$,
4. $\bot = (\emptyset, \Omega_q)$.

In the example context of Figure 1, the most interesting graph concept has as an extent the set of triples (president, city, state): $\{(Obama, Honolulu, Hawaii), (Clinton, Hope, Arkansas)\}$. Its intent is the PGP $((p, c, s), \{((USA, p), president), ((p, c), birthPlace), ((USA, s), state), ((s, c), city)\})$. Other graph concepts are either projections of it (see Section 6), concepts with singleton extents (having one tuple), the top concepts (having all tuples), and the bottom concepts (having no tuple).

6 Projections Between Concept Lattices

In the previous section, we have shown the existence of a family of graph concept lattices, one for each arity $k \geq 0$. This is analogous to previous work with power context families [18], with the important difference that each k-concept has as an intent a PGP that may combine relationships of different arities. As FCA lattices are generally used as search spaces for knowledge discovery, it is useful to relate concepts from different lattices, i.e. having different arities. To that purpose, we use *projections*, a fundamental operation in relational algebra (see Section 2 for definitions and notations). A projection enables to permute, duplicate, and remove *columns* in relations and PGPs. Our projections differ from those of pattern structures, which are used to simplify graph patterns [9].

We first demonstrate that the set of all concept extents is closed by projection because the projection of the extension of a PGP is the extension of the projection of the PGP.

Lemma 4. *For all $Q \in \mathcal{Q}_k$, and $\pi \in \Pi_k^l$, we have: $\pi(ext(Q)) = ext(\pi(Q))$.*

Proof. $\pi(ext(Q)) = \pi(\{\overline{o} \mid \exists \sigma : \sigma(\overline{x}) = \overline{o} \wedge \sigma(P) \subseteq I\})$
$= \{\pi(\overline{o}) \mid \exists \sigma : \sigma(\overline{x}) = \overline{o} \wedge \sigma(P) \subseteq I\} = \{\overline{o}' \mid \exists \sigma : \pi(\sigma(\overline{x})) = \overline{o}' \wedge \sigma(P) \subseteq I\}$
$= \{\overline{o}' \mid \exists \sigma : \sigma(\pi(\overline{x})) = \overline{o}' \wedge \sigma(P) \subseteq I\} = ext((\pi(\overline{x}), P)) = ext(\pi(Q))$ ■

Theorem 4. *Let $\pi \in \Pi_k^l$ be a projection. For every k-concept (R, Q), $(\pi(R), int(ext(\pi(Q))))$ is a l-concept. The latter is called the π-projection of concept (R, Q), denoted by $\pi(R, Q)$.*

Proof. From Lemma 4, we have $\pi(R) = \pi(ext(Q)) = ext(\pi(Q))$, so that $\pi(R)$ is a l-concept extent. The corresponding l-concept intent is $int(\pi(R)) = int(\pi(ext(Q))) = int(ext(\pi(Q)))$. ■

For example, let $P = \{((x), country), ((x, y), president)\}$ be a graph pattern relating a country to its president. The PGP $Q = ((x, y), P)$ returning pairs (country,president) can be projected to the PGP $((y), P)$ returning all presidents of a country, or to the PGP $((x), P)$ returning all countries having a president. In the particular case where $k = l$, the two concepts belong to the same lattice. For example, the PGP $((y, x), P)$ is a permutation of Q. Therefore, there may be up to $k!$ permutations of a single k-concept in the same concept lattice. As those permutations are equivalent from the point of view of knowledge discovery, a concept lattice could in principle be made smaller by retaining only one of the permutations. Note that a concept can sometimes be equal to some of its permutations. For example, The query $((x, y), \{((x, z), parent), ((y, z), parent)\})$, which defines the sibling relationship, has the same extension as its permutation (y, x). This equality comes from a symmetry in the PGP.

The existence of a projection between two concepts defines a pre-ordering $\leq_\pi$ on the set of all concepts $\mathcal{C} = \bigcup_{k \geq 0} \mathcal{C}_k$. Indeed, it satisfies transitivity (by composing projections), reflexivity (by using the identity projection), but not antisymmetry (consider a permutation and its inverse). Two concepts are then

equivalent ($=_\pi$) if they are a permutation one of the other. The example concept from Section 5.3, which contains triples (president, city, state), has 6 distinct permutations, and 2×3 projections of arity 2, and 3 projections of arity 1. The fact that all those projections have the same number of instances as the example concept reveals functional dependencies from any column to the others. For instance, the president determines the city and state. The functional dependency from state to president would be violated if two presidents in the concept extent were associated to the same state. This example suggests that the partial ordering $\leq_\pi$ can support the discovery of functional dependencies, and may generalize previous work on multi-valued contexts [1].

7 Conclusion and Future Work

We have proposed an extension of FCA, G-FCA, where objects are replaced by tuples of objects. In G-FCA, the context is a *knowledge graph*, concept intents are *projected graph patterns (PGP)*, and concept extents are *object relations*. A set-like algebra of PGPs is defined with inclusion, intersection, and union. PGP inclusion is related to graph matching, and hence to query answering. PGP intersection is related to finding common subgraphs under homomorphism, and hence to data mining and machine learning. The constructive definitions of PGP operations already allow for a direct implementation, but more efficient algorithm have to be devised for practical use. Another objective is to clarify the relationship between G-FCA and previous FCA works, notably power content families and concept graphs [18], Relational Concept Analysis [16], $\mathcal{EL}$-implication bases [3], and concept lattices of relational structures [12].

The results presented in this paper have yet a limited utility, and it remains to show how FCA applications can be transposed to G-FCA. The most common application is to compute and visualize concept lattices. The main difficulty is the huge number of graph patterns, even closed ones [19]. Restrictions can be applied to PGPs (e.g., fully connected patterns, bounded arity, graph isomorphism), but those will probably not be enough in practice due to the combinatorial explosion of graph patterns. Another common application is the discovery of implication rules. In G-FCA, this would correspond to unsupervised ILP, but limited to exact rules. Given how costly ILP is in the supervised setting, the computation of all implication rules could be a challenge. Alternately, those implications could be computed on the need, specifically for each (tuple of) object(s) to be classified [8,14]. Yet another application is to use the concept lattice as a search space for information retrieval. In fact, we have already formalized and implemented such an application [6], and it was the inspiration for the current work.

References

1. Allard, P., Ferré, S., Ridoux, O.: Discovering functional dependencies and association rules by navigating in a lattice of OLAP views. In: Kryszkiewicz, M., Obiedkov, S. (eds.) Concept Lattices and Their Applications. pp. 199–210. CEUR-WS (2010)

2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Baader, F., Distel, F.: A finite basis for the set of EL-implications holding in a finite model. In: Medina, R., Obiedkov, S.A. (eds.) Int. Conf. Formal Concept Analysis. pp. 46–61. LNCS 4933, Springer (2008)
4. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about datalog (and never dared to ask). IEEE Trans. Knowl. Data Eng. 1(1), 146–166 (1989)
5. Chein, M., Mugnier, M.L.: Graph-based knowledge representation: computational foundations of conceptual graphs. Advanced Information and Knowledge Processing, Springer (2008)
6. Ferré, S.: Conceptual navigation in RDF graphs with SPARQL-like queries. In: Kwuida, L., Sertkaya, B. (eds.) Int. Conf. Formal Concept Analysis. pp. 193–208. LNCS 5986, Springer (2010)
7. Ferré, S., Ridoux, O.: A logical generalization of formal concept analysis. In: Mineau, G., Ganter, B. (eds.) Int. Conf. Conceptual Structures. pp. 371–384. LNCS 1867, Springer (2000)
8. Ferré, S., Ridoux, O.: The use of associative concepts in the incremental building of a logical context. In: U. Priss, D. Corbett, G.A. (ed.) Int. Conf. Conceptual Structures. pp. 299–313. LNCS 2393, Springer (2002)
9. 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)
10. Ganter, B., Wille, R.: Formal Concept Analysis — Mathematical Foundations. Springer (1999)
11. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
12. Kötters, J.: Concept lattices of a relational structure. In: Pfeiffer, H., et al. (eds.) Int. Conf. Conceptual Structures for STEM Research and Education, pp. 301–310. LNAI 7735, Springer (2013)
13. Kuznetsov, S.O., Samokhin, M.V.: Learning closed sets of labeled graphs for chemical applications. In: Kramer, S., Pfahringer, B. (eds.) Int. Conf. Inductive Logic Programming. pp. 190–208. LNCS 3625, Springer (2005)
14. Kuznetsov, S.: Fitting pattern structures to knowledge discovery in big data. In: Cellier, P., Distel, F., Ganter, B. (eds.) Int. Conf. Formal Concept Analysis, pp. 254–266. LNAI 7880, Springer (2013)
15. Muggleton, S., Raedt, L.D.: Inductive logic programming: Theory and methods. Journal of Logic Programming 19,20, 629–679 (1994)
16. 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)
17. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: *et al.*, P.M. (ed.) The Semantic Web (ISWC). pp. 245–260. LNCS 8796, Springer (2014)
18. Wille, R.: Conceptual graphs and formal concept analysis. In: Int. Conf. Conceptual Structures. pp. 290–303. LNCS 1257 (1997)
19. Yan, X., Han, J.: Closegraph: mining closed frequent graph patterns. In: ACM Int. Conf. Knowledge discovery and data mining (SIGKDD). pp. 286–295. ACM (2003)