Contribution to the Classification of Web of Data based on Formal Concept Analysis

Justine Reynaud, Yannick Toussaint, Amedeo Napoli

Contribution to the Classification of Web of Data based on Formal Concept Analysis

Justine Reynaud[1,2], Yannick Toussaint[1,2], and Amedeo Napoli[1,2]

[1] INRIA Nancy-Grand Est, 54600 Villers-les-Nancy, F-54600, France
[2] Université de Lorraine, LORIA, UMR 7503, Vandoeuvre-les-Nancy, F-54506, France

Abstract. During the last decade, the web has taken a huge importance in everyday life, and has become what is commonly called a web of data. The available resources can be used by human agents but also by software agents, as it is the case for very large ontologies such as YAGO or resources such as DBpedia. These particular datasets can be linked together for constituting the Linked Open Data (LOD) cloud, where basic data are expressed as (subject, predicate, object) triples. One issue of main interest is knowledge discovery within LOD, which can help information retrieval and knowledge engineering. Formal concept analysis (FCA), which is a mathematical theory allowing classification and data analysis, was already used to classify LOD elements. In this research work, we are interested in analyzing the different approaches (extensions) based on FCA for knowledge discovery in the web of data. One objective is to study the efficiency and the applicability of the existing approaches and to propose some improvements.

Keywords: Formal Concept Analysis, Pattern Structures, Triadic Concept Analysis, Linked Open Data

1 Introduction

In this paper, we would like to extend and complete a previous work on the classification of Linked Open Data (LOD) [2]. The basic unit of LOD is the RDF triple which is composed of three elements, namely a subject, a predicate and an object, i.e. *(subject, predicate, object)*. It can be noticed that subjects, predicates and objects can be organized into a partial ordering depending on a specific schema (i.e. RDFS) or an ontology (e.g. YAGO or DBpedia Ontology).

The classification of LOD should take into account the RDF triple as a basic unit and this can be done in several ways. We distinguish ways that relies on a "triples" view of LOD and ways that relies on a "graph" view. Following the lines of [2], we consider an approach which considers triples and which is based on pattern strucures [6]. We complete this approach by integrating the organization of predicates in the classification of RDF triples.

In [2], it was shown how the classification of RDF triples amounts to classify pairs of objects and attributes –as in a binary context– where attributes are partially ordered. Actually, given a subject which corresponds to an object

in a binary context, predicates are considered "one by one" and attributes are viewed as "ranges" of the predicates. Attributes are organized within a partial ordering and there are two main ways of dealing with this order. The first one is to consider a "scaling" as in [3,4] where the description of an attribute correspond to the set of its ancestors in the attribute hierarchy, and the similarity between two attributes is given by the minimal elements in the intersection of their descriptions In the second way, it can be shown that pattern structures for structured sets of attributes are very well adapted to solve the problem of classifying RDF triples for analyzing the content of LOD. As it was precised in [6], it can be considered that the description of an attribute is an antichain and the similarity is given by the intersection of antichains (which is actually an alternative way of handling the scaling introduced just above).

Both approaches are discussed in [2], where in addition a specific procedure based on RMQ is used for efficiently computing the intersection of antichains. In the present work we add the "predicate dimension" within RDF triples and we propose first elements for extending our preceding approach by considering the predicate classification. The basics of the approach are detailed but for the moment no experiments are proposed, which is planned in a future work. However, we show that this new proposal is sound and formally consistent.

The paper is organized as follows. First, we present some preliminaries about the web of data and LOD. Then, we recall and discuss the previous approaches for classifying RDF triples or RDF graphs. Finally, we detail and illustrate our new proposal, and prove a main proposition on the similarity of object descriptions.

2 Web of data

Basically, the web of data consists of resources and relations between those resources. It can be represented as a graph where nodes are resources and edges are relations.

The core of the web of data is the `RDF` (Resource Description Framework) language, based on graph model where basic units are (`subject`, `predicate`, `object`) triples. These triples, also called `RDF` statements, describe facts [1].

Definition 1 (RDF Triple). *Given a set of URIs U, blank nodes B and literals L, an RDF triple is represented as* $t = (s,p,o) \in (U \cup B) \times U \times (U \cup B \cup L)$*, where:*

- s *is called subject,* p *predicate and* o *object;*
- *U is the set of all resources identified by a* `URI` *(Universal Resource Identifier);*
- *B is the set of resources that are unidentified (called* blank node*);*
- *L is the set of literals, which are values like strings, dates or integers.*

As its name suggests, `RDF` allows one to describe *resources*. Resources can refer to any object or thing. For convenience, the terms borrowed from description logics can be used to distinguish different types of resources:

Properties: express binary relationships between any two entities;

Classes: represent sets of entities;

Instances: entities that belong to a class;

Variables: unidentified resources (i.e. blank nodes).

`RDF` has some specific vocabulary. For example, the property `rdf:type` enables to declare an instance as belonging to a class. This expression has two parts, separated by a colon. The second part identifies the specific resource of this vocabulary, here "type". The first part, also called *prefix*, is an abbreviation for "`http://www.w3.org/1999/02/22-rdf-syntax-ns#`", the *namespace* which refers to the `RDF` vocabulary. Each vocabulary has its namespace.

In order to give `RDF` some structure, schemas are used. Schemas describe constraints on facts. They correspond to the TBox in description logics terms. Each vocabulary may have its own schema, that is a structure between its entities, called its reference schema. The structure of `RDF` triples is based on `RDFS` (`RDF` Schema), bringing additionnal properties such as `rdfs:subPropertyOf` and `rdfs:subClassOf`. These two additionnal properties enable to build a hierarchy of relations one the one hand and classes on the other hand. Afterwards, these properties will be denoted `subP` and `subC`, respectively. Instances can be linked to the hierarchy of classes, but they are not part of it, since they are related to a class with `rdf:type`. Thus, in the following, a hierarchy will refer to an orderred set of classes, and instances will be attached to their class (for simplicity, a unique class per instance is considered here).

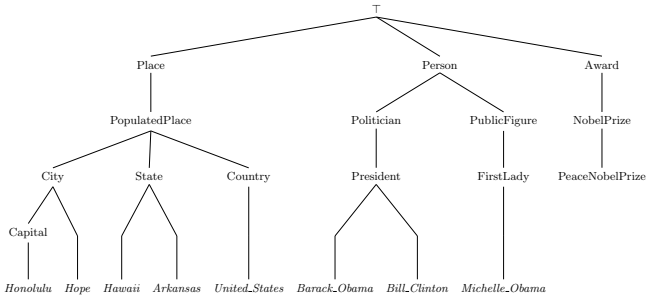
The language `SPARQL` offers to run queries on the web of data. A query is composed of `RDF` triples containing variables. For example, the query `SELECT ?x WHERE {?x rdf:type C}` returns all the instances of C.

A toy knowledge base, freely inspired by the example of S. Ferré in [5], is presented in Figure 1. The Figures 1b and 1d correspond to a set of `RDF` statements and the associated graph respectively. The Figures 1c and 1a represent the background knowledge. Instances of the knowledge base are considered and linked to the class hierarchy.

Reference schemas. Resulting from the linked open data, each data set is connected to others. Thus, hierarchies are not guarantee to be a tree – instances belonging to classes of their schema and to classes of another schemata for example.

In this work, we consider that all the classes on one hand and all the properties on the other hand belong to the same reference schema; that is, have the same namespace. We will also assume that, for any reference schema, there are neither `subP` nor `subC` cycle.

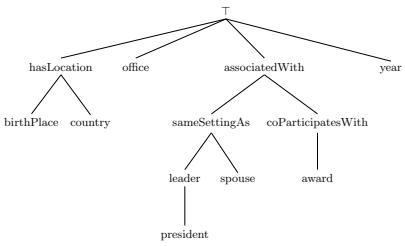
Another concern is that, the tree structure is not guaranteed, even if there is no cycle. If C_1 and C_2 are two incomparable classes, both subclasses of C_3 and C_4 that are also incomparable, then, we lost the tree structure and we have a conflict. Here, we suppose that we know how to linearize the hierarchy of each class as the linearization of a product of two partial orderings.



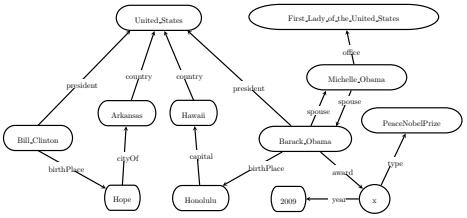
(a) `subC` relation

```
Barack_Obama president United_States .
Bill_Clinton president United_States .
Barack_Obama birthPlace Honolulu .
Bill_Clinton birthPlace Hope .
Honolulu capital Hawaii .
Hope city Honolulu .
Arkansas country United_States .
Hawaii country United_States .
Barack_Obama spouse Michelle_Obama .
Michelle_Obama office First_Lady_of_the_United_States .
Michelle_Obama spouse Barack_Obama .
Barack_Obama award x .
x type PeaceNobelPrize .
x year 2009 .
```

(b) Knowledge base as `RDF` triples



(c) `subP` relation



(d) Knowledge base as graph

Fig. 1: **Toy knowledge base.** The subfigures 1d and 1b illustrate a set of facts (this corresponds to the ABox in description logics terms). Subfigure 1c illustrates the hierarchy of properties w.r.t. the `subP` relation. The subfigure 1a shows the hierarchy of classes with their instances.

3 Formal Concept Analysis

Formal Concept Analysis (FCA) [7] is a mathematical framework used for classification and knowledge discovery. As a learning process, FCA allows one to build an ordered set of concepts where objects are classified w.r.t. the attributes that they share.

A lot of extensions have been proposed, and some of them can be usefull to deal with WOD. Two approaches are possible: the first consists in considering the `RDF` graph itself whereas the second consists in considering the `RDF` statements.

Classifying the web of data enables to find inconsistencies resulting from the merging of different data sets. It is also a way to discover relationships or implications that are not explicit in any single data set. Moreover, the visual support given by the lattice allows users to easily navigate through hierarchy for exploratory research.

3.1 WOD as a set of statements

The first approach consider WOD as a set of RDF triples. This approach is related to the works of [4], [8], [9] and [2].

Classification w.r.t. background knowledge The WOD can be classified through RDF triples. However, in order to be interesting enough, this approach has to take into account background knowledge such as the classes and properties hierarchies. This problem has been developed in [4]. The authors consider a set of documents as objects and a set of terms as attributes. Terms belong to a thesaurus and they are organized in a tree structure. In order to build the lattice, they consider M^* the set of terms in the thesaurus, and define an order $\leqslant_{M^*}$, meaning that "*any attribute implies any of its more general attributes.*" For example, if the term *indexing* is more specific than *information-analysis* in the thesaurus and if it is an attribute of a document d, then *information-analysis* is also an attribute of d. Then, the authors define the intersection $\cap^*$ of two intents m_1 and m_2 as "*the most specific attributes in* M^* *that are more general than* m_1 *and* m_2." With this new operator, they can build concept lattices taking into account background knowledge. In the following, we will present an extension of this approach.

Triadic Concept Analysis The triadic concept analysis (TCA) [12] considers an additionnal dimension (the modus). This implies a ternary relation $Y \subseteq G \times M \times B$ which can be interpreted as *"the object g takes the value b under the condition m."* The corresponding Galois connections are more complex: they are three and each one corresponds to a dimension expressed in terms of the two others.

Mining triconcepts. TCA is used to describe folksonomies, i.e. communities of users U who can annotate resources R with keywords (tags) T. An example of taxonomy is Bibsonomy, a tool allowing users to manage publications and to tag them. Here a concept would be a group of users tagging a set of resources with identical keywords The algorithm Trias [8] has been proposed to mine tri-concepts. It is based on a projection of the triadic context (U, T, R, Y) onto a dyadic context $(U, (T \times R), I)$. For each concept (A, J) found in $(U, (T \times R), I)$ context, a new dyadic context (T, R, J) is built. For each concept (B, C) found, (A, B, C) is a concept of (U, T, R, Y).

Finding biclusters. In [9], the authors aim to find biclusters in a numerical dataset. The context is similar to a standard formal context, but instead of a binary context, a context with numerical values is considered. In order to have a triadic context, objects and attributes remain the same and the numerical values are transformed into a third dimension by means of an interordinal scaling. A threshold θ is provided and the new dimension is made up of intervals whose range sizes are less or equal to θ.

Limitations. Given that TCA can handle three dimensions, it could be interesting for WOD. However, a simple approach does not take into account

the background knowledge. Thus, taking into account the hierarchies implies a scaling. This implies to add all classes in the dimension of subjects and objects, and all properties in the dimension of predicates. This approach works well in the case of biclustering, when a threshold is given, limiting the number of intervals to consider. By contrast, with WOD, such a threshold can hardly be considered. Moreover, having this constraint does not guarantee the scalability.

Pattern structures Pattern structures (PS) [6] are proposed in order to consider data which are not binary data. Attributes then become descriptions which are partially ordered thanks to a similarity operation. Formally, a pattern structure is defined as follows:

Definition 2 (Pattern structure). *Let G be a set of objects, $(D, \sqcap)$ a semi-lattice and $\delta : G \to D$ a mapping. Then $(G, (D, \sqcap), \delta)$ is called a pattern structure. The new Galois connections are the following:*

- $A^{\square} = \bigsqcap_{g \in A} \delta(g)$ *for* $A \subseteq G$
- $d^{\square} = \{g \in G \mid d \sqsubseteq \delta(g)\}$ *for* $d \in D$

Pattern structures are used on triples in [2]. In this work, the authors aim to provide a navigation space over RDF resources. The underlying idea is that, when a resource is a subject in a triple, we can consider that the pair (p,o) is *describing* this resource. Thus, two resources can be compared regarding how they are described by the triples in which they are subjects. Moreover, there is background knowledge related to objects provided by the properties `rdf:type` and `subC`. Considering a set of triples $\mathcal{B} = (s_i, p_j, o_k)$ and a hierarchy of classes, this idea can be materialized in term of formal concepts. To that purpose, a pattern structure $(G, (D, \sqcap), \delta)$ is constructed as described below.

Entities[3] and their descriptions.

The set $\mathcal{B}$ of `RDF` statements is built with a `SPARQL` query on a specific namespace. The associated reference schema is used to construct the hierarchy of classes from this namespace.

First, resources that are subjects of at least one triple in $\mathcal{B}$ are considered as entities of the pattern structure: $G = \{s_1, \ldots s_n\}$. They will be compared regarding the objects they share for each predicates. Objects can be instances or classes, but here only classes are considered. Indeed, the hierarchy between the objects is based on the property `rdfs:subClassOf`, but instances are linked to classes with the relation `rdf:type`. In order to maintain the consistency, objects that are instances are replaced by the class they belong to.

Each pair (p, o) such that $(s, p, o) \in \mathcal{B}$ is mapped to a description $d \in D$. A description $d \in D$ is a pair (p_i, O_i) where p_i is a predicate and O_i is a set of objects in the range of p_i. Given an entity $s \in G$, its description is defined as follow: $\delta(s) = \{d_1^s, \ldots, d_n^s\}$ where $d_j^s = (p_j, O_j^s)$ with $j = \{1, \ldots, n\}$. Each

[3] The term *object* is ambiguous: it denotes both the first part of a pattern structure and the last element of an `RDF` triple.The term *entity* will be used to denote objects of the pattern structure. The term *object* remains for the triples.

elementary description (p_i, o_j) is replaced with $(p_i, C(o_j))$ where $C(o_j)$ is the class of o_j in the reference schema.

Given two descriptions $\delta(x) = \{d_1^x, \ldots, d_n^x\}$ and $\delta(y) = \{d_1^y, \ldots, d_m^y\}$, we have $\delta(x) \sqsubseteq \delta(y)$ if $\forall d_i^x \in \delta(x), \exists d_j^y \in \delta(y)$ s.t. $i = j$ and $\forall o_i^x \in O_i^x, \exists o_j^y \in O_j^y$ s.t. $C(o_i^x)$ `subC` $C(o_j^y)$.

Similarity between descriptions. The hierarchy of classes from the reference schema is considered. Thus, the similarity between two objects is defined as follows:

$$\begin{aligned}\delta(x) \sqcap \delta(y) =& \{d_1^x, \ldots, d_n^x\} \sqcap \{d_1^y, \ldots, d_m^y\} \\ =& min \bigcup_{\substack{i \in \{1,\ldots,n\} \\ j \in \{1,\ldots,m\}}} \{d_i^x\} \sqcap \{d_j^y\}\end{aligned}$$

where min returns the minimal elements

$$\{d_i^x\} \sqcap \{d_j^y\} = \begin{cases} lcs(C(o_i^x), C(o_j^y)) & \text{if i = j} \\ \emptyset & \text{else} \end{cases}$$

where lcs returns the most specific superclass

This definition is close to the $\cap^*$ used in [4]. The $\sqcap$ operation between two descriptions corresponds to finding the most specific attribute in M^*. Considering only the minimal of the union corresponds to "*retaining only the most specific elements of the set generated this way*".

Limitations. This work introduces a method to mine triples with pattern structures as in [2]. The main limitation is that, the hierarchy of predicates is not considered.

3.2 WOD as a graph

Instead of considering `RDF` triples, it is possible to consider the associated graph. This is what is done in some approaches like [11], [10] and [5].

Pattern structures for graphs In [11], pattern structures are used to classify graphs. Each graph is considered as an object and the set of all its subgraphs is considered as the description. Thus, the similarity between two graphs is the set af all the subgraphs they have in common.

As this approach is expensive, graphs can be simplified by the mean of projections. That is, instead of considering all the subgraphs of a graph, the description is something simpler like the set of chains of a certain size that compose the graph.

Concept lattices of conceptual graphs In [5], an extension to FCA for conceptual graphs, called G-FCA, is proposed. Compared to RDF graphs, conceptual graphs (CG) are oriented bipartite graphs. The two kinds of nodes are

classes in one hand and relations in the other hand. Contrary to RDF graphs which consider only binary relations, CGs handle n-ary relations. *Projected graph patterns* are introduced as concepts. It is similar to a SPARQL query where the graph query is the intent and the candidate solutions are the extent.

Example 1. Given the Figure 1, we have the concept *({(Hope,Arkansas), (Honolulu,Hawaii)}, {(?p, birthPlace, ?x), (?x, city, ?y)})*. Note that, the syntax is different from [5], since the example is adapted to `RDF`. This concept is associated to the query `SELECT ?x ?y WHERE { ?p birthPlace ?x . ?x city ?y.}`

Concept lattices of `RDF` graphs In [10], in addition to the `RDF` graph, a formal context corresponding to the background knowledge is considered. This context has a set of resources that are objects and attributes at the same time. Considering one resource as object, its attributes are the set of resources that are "more general" regarding `subC` and `subP` properties.

Example 2. Given the knowledge base Figure 1, a part of the formal context could be the following:

	City	Capital	president	sameSettingAs	*Honolulu*	*Hawaii*
City	×					
Capital	×	×				
president			×	×		
sameSettingAs				×		
Honolulu	×	×			×	
Hawaii						×

The extent of the pattern is a set of resources whereas the intent is a triple graph. A triple graph is basically a subgraph and some background knowledge. A morphism between two triples graphs is defined and corresponds to an order on intents. Moreover, a product between triple graphs is defined such that the join of two concepts (i.e. triple graphs) corresponds to their product.

Example 3. Given the knowledge context and the graph pattern corresponding to the triple (Honolulu,capital,Hawaii), the graph corresponding to the triple (Honolulu,city,Hawaii) is more general.

4 Taking into account the three parts of the triple

In this section, we propose a generalization of [2] : instead of considering subjects described by (predicate,object) pairs, we consider the entire triple as a description.

Entities and descriptions We suppose that each triple has a unique identifier, like a *transaction id*. These identifiers are the entities of the pattern structure : $G = \{t_1, \ldots, t_n\}$. The description of an entity is a mapping to the triple itself, where instances are replaced by the class they belong. As to not complefy the notation, $C(s)$ and $C(o)$ will be written s and o.

Order on descriptions Given the descriptions $\delta(t_i) = (s_i, p_i, o_i)$ and $\delta(t_j) = (s_j, p_j, o_j)$, the partial order on descriptions is defined as follow:

$$\delta(s_i, p_i, o_i) \sqsubseteq \delta(s_j, p_j, o_j) \Leftrightarrow s_j \texttt{ subC } s_i, p_j \texttt{ subP } p_i, o_j \texttt{ subC } o_i$$

Similarity between descriptions The similarity between two triples t_i and t_j is defined as follow:

$$\delta(t_i) \sqcap \delta(t_j) = (lcs_c(s_i, s_j), lcs_p(p_i, p_j), lcs_c(o_i, o_j))$$

Proposition 1. $\delta(t_i) \sqsubseteq \delta(t_j) \Leftrightarrow \delta(t_i) \sqcap \delta(t_j) = \delta(t_i)$.

Proof.

$$\begin{aligned}
\delta(t_i) \sqsubseteq \delta(t_j) &\Leftrightarrow s_j \texttt{ subC } s_i, p_j \texttt{ subP } p_i, o_j \texttt{ subC } o_i \\
&\Leftrightarrow lcs_c(s_i, s_j) = s_i, lcs_p(p_i, p_j) = p_i, lcs_c(o_i, o_j) = o_i \\
&\Leftrightarrow (lcs_c(s_i, s_j), lcs_p(p_i, p_j), lcs_c(o_i, o_j)) = (s_i, p_i, o_i) \\
&\Leftrightarrow (s_i, p_i, o_i) \sqcap (s_j, p_j, o_j) = (s_i, p_i, o_i) \\
&\Leftrightarrow \delta(t_i) \sqcap \delta(t_j) = \delta(t_i)
\end{aligned}$$

Similarity between set of triples The similarity of two triples can be generalized to set of triples. The description of a set of triple T is the set of descriptions of each of its triples : $\Delta(T) = \{\delta(t) \mid t \in T\}$.

Given two sets of triples T_1 and T_2, the similarity $T_1 \sqcap T_2$ is the set of minimal triples $\delta(t_i) \sqcap \delta(t_j)$ for all t_i in T_1 and for all t_j in T_2 given the order on descriptions.

5 Conclusion

In this paper we discussed some approaches for dealing with the classification of web of data, and more precisely of sets of RDF triples, i.e. *(subject, predicate, object)*. Actually, this kind of classification process is based on the classification of pairs *(subject, attribute)* which simulate the RDF triples and where attributes correspond to object triples and are structured within a hierarchy. Here, we proposed an extension to a previous work which takes into account the classification of predicates which was not the case before. We gave a formal presentation of this proposal and for future work we are planning to make a series of experiments which should (hopefully) validate the current approach.

Acknowledgments

Justine Reynaud is preparing her PhD Thesis with the support of "Région Lorraine" and "Délégation Générale de l'Armement".

References

1. Serge Abiteboul, Ioana Gabriela Manolescu Goujot, and Philippe Rigaux. *Web data management*. Cambridge University Press, New York, 2012.
2. Mehwish Alam, Aleksey Buzmakov, Amedeo Napoli, and Alibek Sailanbayev. Revisiting Pattern Structures for Structured Attribute Sets. In Sadok Ben Yahia and Jan Konecny, editors, *The Twelth International Conference on Concept Lattices and their Applications – CLA 2015, Clermont-Ferrand, France*, pages 241–252. LIMOS Clermont-Ferrand, 2015. CEUR Workshop Proceedings 1466 `http://ceur-ws.org/Vol-1466`.
3. Claudio Carpineto and Giovanni Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2):95–122, 1996.
4. Claudio Carpineto and Giovanni Romano. *Concept Data Analysis: Theory and Applications*. John Wiley & Sons, Chichester, UK, 2004.
5. Sébastien Ferré. A Proposal for Extending Formal Concept Analysis to Knowledge Graphs. In *Formal Concept Analysis*, volume LNCS 9113, pages 271–286, Nerja, Spain, June 2015.
6. Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. In *ICCS*, LNCS 2120, pages 129–142. Springer, 2001.
7. Bernhard Ganter and Rudolf Wille. *Formal concept analysis - mathematical foundations*. Springer, 1999.
8. Robert Jäschke, Andreas Hotho, Christoph Schmitz, Bernhard Ganter, and Gerd Stumme. TRIAS - an algorithm for mining iceberg tri-lattices. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 907–911, 2006.
9. Mehdi Kaytoue, Sergei O. Kuznetsov, Juraj Macko, and Amedeo Napoli. Biclustering meets triadic concept analysis. *Annals of Mathematics and Artificial Intelligence*, 70(1-2):55–79, feb 2014.
10. Jens Kötters. Concept lattices of rdf graphs. *Formal Concept Analysis and Applications FCA&A 2015*, page 81, 2015.
11. Sergei O. Kuznetsov. *Computing Graph-Based Lattices from Smallest Projections*, pages 35–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
12. Fritz Lehmann and Rudolf Wille. *A triadic approach to formal concept analysis*, pages 32–43. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.