Comparing Relational Concept Analysis and Graph-FCA on their Common Ground

Vanessa Fokou, Peggy Cellier, Xavier Dolques, Sébastien Ferré, Florence Le Ber

Comparing Relational Concept Analysis and Graph-FCA on their Common Ground *

Vanessa Fokou[1 [0009−0009−3778−2708]], Peggy Cellier[2 [0000−0002−1495−2534]], Xavier Dolques[1 [0000−0002−5579−1714]], Sébastien Ferré[2 [0000−0002−6302−2333]], and Florence Le Ber[1 [0000−0002−2415−7606]]

[1] Université de Strasbourg, ENGEES, CNRS, ICube UMR 7357, F 67000 Strasbourg
vfokou@unistra.fr, dolques@unistra.fr, florence.leber@engees.unistra.fr
[2] Univ Rennes, CNRS, Inria, IRISA - UMR 6074, Rennes
prenom.nom@irisa.fr

Abstract. Relational Concept Analysis (RCA) and Graph-FCA (GCA) have been defined as Formal Concept Analysis (FCA) extensions for processing relational data and knowledge graphs respectively. Nevertheless, while their purposes and results seem similar, the data modelling and the definition of concepts are different. In this paper, we compare these two approaches on a common basis, considering only unary and binary relations for GCA and the existential quantifier for RCA. We focus on examples showing the similarities and dissimilarities between both methods, and highlighting how cycles are processed differently by RCA and GCA.

Keywords: Relational Concept Analysis · Graph-FCA· Formal Concept Analysis · Relational Data · Cycles.

1 Introduction

With the digitization of human activities, more and more complex and multi-relational data are available for analysis in many fields (e.g., agriculture, transport). In addition, with the rise of semantic web, more and more data on the web is represented in the form of knowledge graphs (e.g., RDF graphs [8], Conceptual Graphs [17]) providing a flexible representation of complex data as a set of interconnected entities with binary, and possibly n-ary relationships. Appropriate methods are therefore necessary to analyze these types of data. Formal Concept Analysis (FCA) [6], a mathematical clustering method widely applied in many fields (e.g., transcriptomic data [1], information science [15]), has shown interesting results for discovering conceptual structures in tabular data. This has led to extensions of FCA to relational data, notably Relational Concept Analysis (RCA) [16] and Graph-FCA (GCA) [3, 5]. These approaches aim to extract conceptual structures in multi-relational data, in the same way that FCA extracts conceptual structures in tables.

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

On the one hand, RCA allows the processing of binary relationships linking several tables and to extract interrelated unary concepts (whose extents are sets of objects). On the other hand, the aims of Graph-FCA (GCA) is the extraction of graph patterns in knowledge graphs. Its allows the processing of n-ary relationships and the extraction of n-ary concepts (whose extents are sets of n-tuples of objects). Nevertheless, both RCA and GCA deal with relational data, and their results seem somehow similar [13, 4]. It would thus be interesting to broaden their comparison, to help an analyst choose which approach to use, depending on the data, or the expected result.

In this paper, we compare the two approaches on a common basis, i.e. applied on binary relations and using the existential quantifier for RCA. Furthermore, since GCA implicitly integrates inverse relationships, we include inverse relationships in the data processed by RCA. The equivalence of RCA and GCA results is studied with two examples, and the role of data cycles in the respective results of the two methods is highlighted. The rest of the paper is organized as follows. Section 2 presents the related work. Principles of FCA, RCA and GCA are described in Section 3 and the comparison of RCA and GCA is detailed in Section 4. Section 5 concludes the paper and discusses future work.

2 Related Work

Several extensions of FCA have been proposed to handle relational data. Triadic Concept Analysis (TCA) [12] was proposed to deal with situations where *"the object g has the attribute m under the condition b"*, this can be considered to deal with any ternary relationship. The result is a concept trilattice where, each triadic concept is a triple made of an extent, an intent and a modus respectively. A power context family [18] is made up of a set of formal contexts with different arities i.e., a context of objects, a context of pairs of objects, a context of triples of objects, etc. A concept lattice is computed for each context, independently of other contexts. In [11], the author generates concept lattices directly from a relational structure, where relational structures assume the role of formal context in standard FCA. Concept intents are relational structures representing conjunctive queries and concept extents are the result sets of the intent queries, i.e. tables. Relational Concept Analysis (RCA) [16] extends FCA with binary relationships between objects, and defines concepts in an iterative way such that each concept is defined in terms of relationships to concepts of the previous iteration until reaching a fix point. Graph-FCA (GCA) [3, 5] extends FCA to knowledge graphs. In a GCA concept, the intent is a Projected Graph Pattern (PGP), analogous to a conjunctive query; and the extent is the set of results of such a query, i.e. a set of object tuples. Unlike RCA, GCA is not limited to binary relations and unary concepts, it also handles n-ary relationships and n-ary concepts. By restricting to binary relationships and unary concepts, the inputs and outputs of RCA and GCA look similar.

A few works in the literature have addressed the question of links between RCA and Graph-FCA. When using RCA, an important issue for its effective use

is the interpretation of its results, so in [4], the authors propose an equivalent representation of a family of RCA concept lattices as a hierarchy of concept graphs in which each concept belongs to one concept graph, and each concept graph exhibits the relationships between several concepts. In [14], the authors present a framework (RCA-SEQ) for helping experts when exploring sequential qualitative data. Firstly, they apply RCA to a relational context family that encodes the analysed sequential data; secondly the resulting family of concept lattices is summarized into a lattice of graphs, each graph representing a navigation path within the lattices. Besides, a practical comparison (on a real dataset) of TCA, RCA and GCA for modelling indeterminate values in ternary data has been carried out in [9]. In this work, ternary relationships are directly represented in GCA and TCA while RCA models the ternary relationships with two binary relationships. Until now comparisons conducted on these methods have focused on particular modelling points, and no general comparison on their characteristics and the results they produce has been done.

3 Preliminaries

In this section, we present the main notions of FCA, RCA and Graph-FCA used in the rest of the paper.

3.1 Formal Concept Analysis (FCA)

FCA [6] consists of discovering conceptual descriptions from a set of objects described by unary attributes, called a formal context. A formal context is a triple $K = (O, A, I)$ where, O and A are sets of objects and attributes respectively and I is a binary relation between O and A, i.e., $I \subseteq O \times A$. For instance, in Fig. 1 (left, up), tables `Garage`, `Person` and `Car` are examples of formal contexts. The incidence relation I induces two mappings (operators), $int : 2^O \to 2^A$ and $ext : 2^A \to 2^O$ such that $int(X) = \{y \in A \,|\, \forall x \in X,\ (x, y) \in I\}$ and $ext(Y) = \{x \in O \,|\, \forall y \in Y,\ (x, y) \in I\}$. A formal concept is then defined as a pair (X, Y) where $X \subseteq O$, $Y \subseteq A$, $int(X) = Y$ and $ext(Y) = X$; X and Y respectively represent the extent and the intent of the formal concept (X, Y). The subsumption relation on the concepts of a context forms a concept lattice. FCA result on the `Garage` context is the concept lattice presented in Fig. 1 (right).

3.2 Relational Concept Analysis (RCA)

RCA [16] was designed to extend FCA to relational data. Its input data is a Relational Context Family (RCF), which is made up of a collection of formal contexts, one for each category of objects, along with a collection of binary relations connecting the objects of one context to the objects of the same or another context. The six tables in Fig. 1 form a RCF with 3 formal contexts (`Garage, Person` and `Car`) and 3 relational contexts (`Maintain, Sell` and `Owner`). Formally, a RCF is a pair ($\mathbf{K}$, $\mathbf{R}$) where:

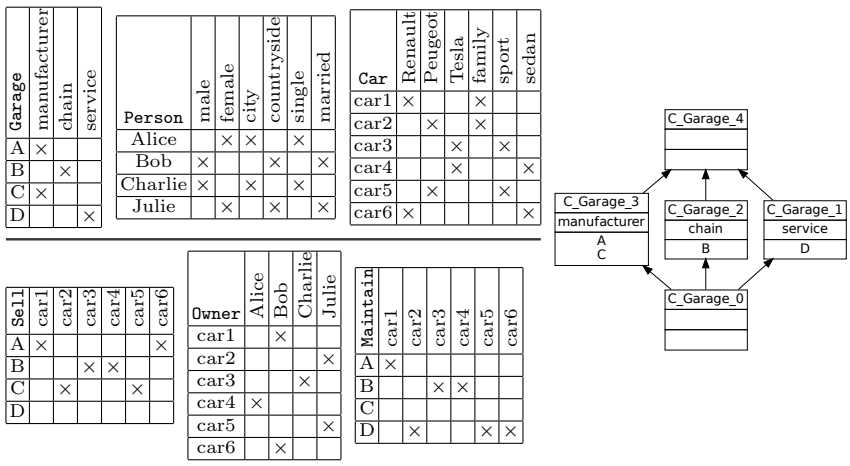

Fig. 1. `Garage-Car-Person` RCF_1 (left) – formal contexts above and relational contexts below – and lattice on `Garage` context (right).

- $\mathbf{K} = \{\mathcal{K}_i\}_{i=1..n}$ is a set of formal contexts $\mathcal{K}_i = (O_i, A_i, I_i)$ and
- $\mathbf{R} = \{r_k\}_{k=1..m}$ is a set of relations r_k where $r_k \subseteq dom(r_k) \times ran(r_k)$, and $dom(r_k), ran(r_k) \in \{O_i\}_{i=1..n}$ are respectively the domain and range of r_k.

For instance, in the running example, we have $\mathbf{K} = \{\texttt{Garage, Person, Car}\}$ and $\mathbf{R} = \{\texttt{Sell, Owner, Maintain}\}$ with $dom(\texttt{Sell}) = dom(\texttt{Maintain}) = O_{\texttt{Garage}}$, $ran(\texttt{Sell}) = ran(\texttt{Maintain}) = O_{\texttt{Car}}$, $dom(\texttt{Owner}) = O_{\texttt{Car}}$ and $ran(\texttt{Owner}) = O_{\texttt{Person}}$. This example is called RCF_1 in the following.

To take full advantage of the information contained in relational contexts, RCA relies on the relational scaling mechanism [16, 2] to capture the information encoded in the object-object contexts through relational attributes. These relational attributes are used to extend formal contexts along the RCA process. In the following, the result of RCA execution on a RCF is denoted by RCA(RCF).

Definition 1 (Relational attribute). *A relational attribute is an expression $qr(C)$, where q is a scaling quantifier ($\exists, \exists\forall, ...$), r is a relationship between two groups of objects and C is a concept whose extent contains objects from $ran(r)$.*

Overall, the RCA process is iterative and each iteration includes 2 steps: (1) lattice construction for each formal context, (2) *relational scaling* and enrichment of formal contexts. The next iteration is based on the results of the previous iteration, and the process ends when the lattices obtained in two successive iterations are equivalent. Finally, given a RCF, RCA(RCF) is a collection of concept lattices interconnected by relational attributes named Concept Lattice

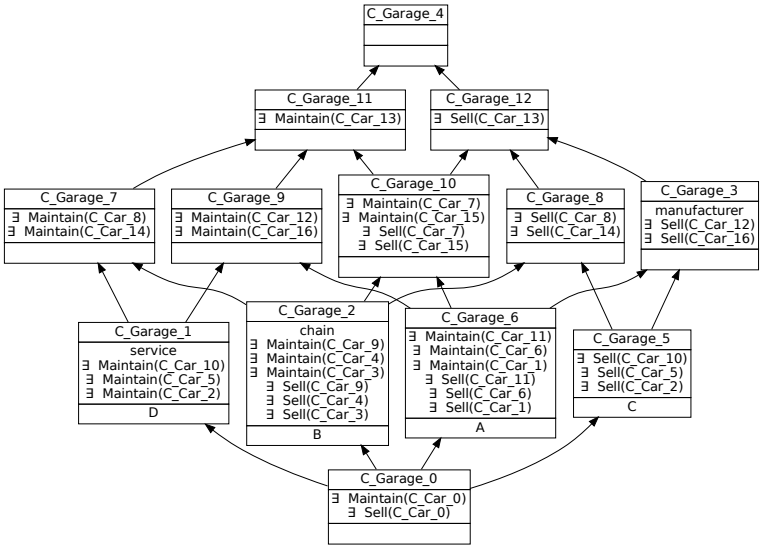

Fig. 2. Concept lattice `Garage` in RCA(RCF_1).

Family (CLF), one concept lattice per formal context. Hence, for the current RCF, we have a CLF made of 3 concept lattices (`Garage, Person` and `Car`). For the sake of space, we only show the concept lattice `Garage` at the end of RCA process as presented in Fig. 2 which is different from the one shown in Fig. 1 (right). Relational attributes are represented in the intents of concepts. For instance, ∃ `Sell(C_Car_10)` is a relational attribute with existential *scaling* linking `C_Garage_5` to `C_Car_10`. As $O_{\texttt{Person}}$ does not constitute the *domain* of any relational context, its concept intents will not contain relational attributes. More details about relational scaling mechanism and relational attributes can be found in [16, 2].

3.3 Graph-FCA (GCA)

To process multi-relational data, GCA [3, 5] adopts a graph point of view, seeing objects as *nodes*, relationships as *directed edges* between nodes, and attributes as node and edge *labels*. The input of GCA is a *graph context* which 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 $\overline{o} \in O^*$ and attributes $a \in A$. An incidence $((o), a)$ describes object o by attribute a like in FCA, while an incidence $((o_1, o_2), a)$ relates object o_1 to object o_2 through the binary relation a like in RCA. An incidence $((o_1, \ldots, o_n), a)$ represents an n-ary relationship. By extension, we call *unary relations* the classical attributes that describe objects. Figure 3 shows the *graph context* named GC_1 corresponding to RCF_1 presented in Fig. 1 (left). Nodes are represented by rectangles where the first compartment identifies the object and the second compartment lists its descriptors, i.e. unary

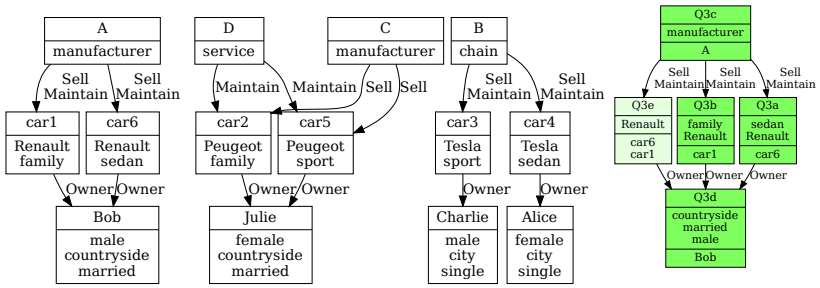

Fig. 3. Graph context (GC_1) of RCF_1.

Fig. 4. An example of GCA pattern.

Table 1. A tabular representation of GC_1 binary relations.

(Garage, Car)	Maintain	Sell
(A, car1)	×	×
(A, car6)	×	×
(D, car2)	×	
(D, car5)	×	
(C, car2)		×
(C, car5)		×
(B, car3)	×	×
(B, car4)	×	×

(Car, Person)	Owner
(car1, Bob)	×
(car6, Bob)	×
(car2, Julie)	×
(car5, Julie)	×
(car3, Charlie)	×
(car4, Alice)	×

relations/attributes. Unary relations are used to label nodes while ($n > 1$)-ary relations are used to label edges connecting n nodes as presented in Table 1 which shows the binary relations of GC_1. For example, the object `A` (garage) is described as a `manufacturer` and is connected to the object `car1` through the binary relations `Sell` and `Maintain` (see Fig. 3). This is formally expressed as: {`((A, car1), Sell), ((A, car1), Maintain), ((A), manufacturer)`}. In the following, $GCA(GC_1)$ expresses the result of Graph-FCA on GC_1.

Graph-FCA defines the extent of a *graph concept* as a set of n-tuples of objects and the intent as a Projected Graph Pattern (PGP) which expresses what those n-tuples of objects have in common; n is the arity of the concept. Those notions are defined in the following.

Definition 2 (Graph pattern and PGP). *Let $\mathcal{V}$ be an infinite set of variables. A graph pattern $P \subseteq \mathcal{V}^* \times A$ is a set of n-ary edges with variables as nodes, and attributes as labels. A PGP is then a couple $Q = (\overline{x}, P)$ where P is a graph pattern and $\overline{x} \in \mathcal{V}^*$ is called the projection tuple.*

A graph pattern is similar to a small graph context, except that objects are abstracted into variables. Figure 4 shows an example of graph pattern `P3` with `a-e` as variable names. A PGP is simply a graph pattern with focus on one or several nodes. For instance, `Q3e` = (`e`), `P3`) is the unary PGP with focus on node `e`. A PGP can be seen as a conjunctive query, whose results are tuples

of objects that can replace the projected variables and match the pattern over the graph context. PGPs form a bounded lattice modulo PGP equivalence. A PGP is more specific than (or equivalent to) another PGP if the results of the former is necessarily a subset of the results of the latter, whatever the graph context [5]. The third compartment of a node (Fig. 4) represents the results for that projected node as a list of objects. For instance, `Q3e` has two results: `car1` and `car6`. A GCA concept is a pair (R, Q) such that Q is a PGP (concept intent), R is the set of results (concept extent) of Q, and Q is the most specific PGP for those results. In this paper we only consider unary concepts, in which there is only one projected node in concept intents. Figure 4 therefore shows 5 distinct unary concepts, 1 garage concept, 3 car concepts, and 1 person concept. Each concept is referenced by its intent PGP, e.g. `Q3e`, and its extent can be found in the third compartment of the projected node. The nodes `a-d` (brightly-colored) belong to the *pattern core*, i.e. its minimal retract. A retract of a graph G is a subgraph H of G such that G is homomorphic to H (see [7], p. 112). When the projected node x is a core node, the non-core nodes can be ignored in the pattern of PGP Qnx because they are redundant with core nodes. However, when the projected node is not a core node, it must of course be included in the pattern of the PGP, possibly along with other non-core nodes. The latter nodes are then indicated in parentheses in the first compartment of node boxes. For instance, node `e` is only used for PGP `Q3e`.

The GCA unary concepts are organized into a single lattice, in contrast to the family of concept lattices in RCA. However, as the GCA output hides both top and bottom concepts for the sake of readability, this output appears as a family of concept hierarchies, one for each category of objects, similarly to RCA. The GCA tool provides a lattice view, as an alternative to the pattern view, as shown in Fig. 5 with the `Person` concept lattice. In this view, a $(n > 1)$-relation between two nodes is represented in the description of nodes (second compartment), along with unary relations. For instance, the `Owner` relation between nodes `a` and `d` in `Q3` (Fig. 4) is represented by edge `[Owner a _]` in the description part of `Q3d` (Fig. 5). Note that those edges are to be read as part of one pattern. For instance, edges `[Owner a _]` in `Q3d` and `[Owner a _]` in `Q5c` are distinct. Their generalization in the supremum concept `Q6a` appears as edge `[Owner d _]`.

4 RCA and Graph-FCA Comparison

This section compares both approaches in the following order: a glance at the obvious differences, examples showing the similarities, dissimilarities and how cycles are processed differently in both approaches. Since intents are represented differently in both approaches, the comparison of concepts is done at the extensional representation level; thus, a RCA concept (named r-concept) corresponds to a GCA concept (named g-concept) when they have the same extent. To illustrate our comparison, we rely on the above RCF_1 / GC_1 example.

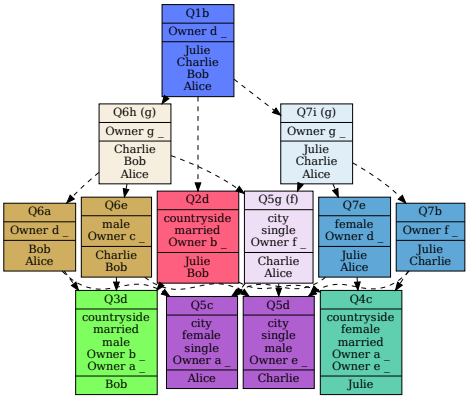

Fig. 5. `Person` concept lattice from $\mathrm{GCA}(\mathrm{GC}_1)$.

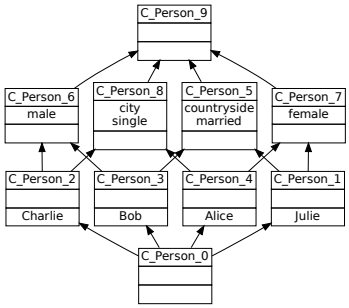

Fig. 6. `Person` lattice from $\mathrm{RCA}(\mathrm{RCF}_1)$.

4.1 A First Glance at the Differences Between RCA and GCA

We first state the obvious differences between RCA and Graph-FCA.

- RCA uses various scaling quantifiers [2] while graph patterns in GCA are implicitly existentially quantified.
- GCA handles n-ary relations, while RCA handles only unary and binary relations.
- GCA handles unary and n-ary concepts, while RCA is limited to unary concepts (we focus on unary concepts in this work).
- RCA concepts are defined in term of relational attributes while GCA concepts are defined in term of graph patterns.
- RCA concepts are computed through a fixed-point iteration while GCA relies on graph intersection.

Furthermore, GCA and RCA do not consider relationships in the same way. GCA automatically considers relationships in both directions, while RCA considers relationships as explicitly defined in the RCF. For illustration, RCF_1 in Fig. 1 (left) and GC_1 in Fig. 3 represent the same information: garages that sell and/or maintain cars owned by a person. However, the definition of the relationships $\texttt{Sell}(O_{\texttt{Garage}}, O_{\texttt{Car}})$, $\texttt{Maintain}(O_{\texttt{Garage}}, O_{\texttt{Car}})$ and $\texttt{Owner}(O_{\texttt{Car}}, O_{\texttt{Person}})$ for RCA means that the `Garage` and `Car` concepts will contain relational attributes in their intent in addition to unary attributes unlike the `Person` concepts which will only have unary attributes in their intent, since the objects of the latter do not constitute the domain of any relational context.

On the GCA side, as in the graph context relations are oriented but can be traversed in both ways, the `Person` concept lattice presented in Fig. 5 contains links to `Car` concepts. For instance, the intent of concept `Q3d` contains the edges

`[Owner a _]` and `[Owner b _]`. This means that instances of concept `Q3d` are characterized by the relation $\texttt{Owner}^{-1}$ to `Car` nodes `a` and `b`. In this example, the concepts produced by RCA on RCF_1 (37 concepts without $\bot$) are included in those produced by GCA on GC_1 (45 concepts without $\bot$)[3], this is expressed by $\text{RCA}(\text{RCF}_1) \subset \text{GCA}(\text{GC}_1)$. The 8 additional g-concepts have to do with inverse relationships.

In the following, to further explore the difference between the two approaches, we will compare them on their common grounds by choosing the following parameters: scaling quantifier $\exists$ for RCA because GCA has only quantifier $\exists$ implicitly, data with only unary and binary relations because n-ary relations ($n > 2$) require a specific modelling in RCA [10]. Finally, inverse relations are included in the RCF for RCA processing.

4.2 Adding Inverse Relations to a RCF

The addition of inverse relations in a RCF makes it possible to bring the RCA results as close as possible to those of GCA. The question is how close these results are. We compare RCA(RCF) and RCA(RFC_r), where RCF_r = RCF + inverse relations and we present an example in which RCA(RFC_r) $\equiv$ GCA(GC) and an example in which RCA(RFC_r) $\not\equiv$ GCA(GC), where GC is the corresponding graph context of RCF. For the running example, this is done by adding 3 relational contexts to RCF_1: $\texttt{Sell\_r}(O_{\texttt{Car}}, O_{\texttt{Garage}})$, $\texttt{Owner\_r}(O_{\texttt{Person}}, O_{\texttt{Car}})$ and $\texttt{Maintain\_r}(O_{\texttt{Car}}, O_{\texttt{Garage}})$ which are respectively the inverse of the relation `Sell, Owner` and `Maintain`. For instance, the $\texttt{Owner\_r}(O_{\texttt{Person}}, O_{\texttt{Car}})$ relationship defines the fact that *a person owns a car*. Then, in the RCA process, the concepts built on the `Car` context, by association with the scaling quantifier and the relation `Owner_r`, will be used to form relational attributes to extend the `Person` context. In the following, this extended RCF_1 is denoted by RCF_1_r.

RCA(RCF) $\subseteq$ RCA(RCF_r). The RCA result obtained on RCF_1_r, is to be compared with the one obtained on RCF_1. Adding inverse relations in a RCF leads to the addition of new relational attributes that lead either to the modification of the intent of existing concepts or to the addition of new concepts. This is consistent with the RCA process, because there are always at least as many concepts in a lattice at iteration i+1 as at iteration i. Formally, RCA(RCF) $\subseteq$ RCA(RCF_r) for any RCF. In our example, the concepts built on RCF_1 (37 concepts) are included in those built on RCF_1_r (45 concepts). Figures 6 and 7 present the `Person` lattices built respectively from RCF_1 and RCF_1_r. There are 4 additional concepts in the second lattice, due to relational attributes of the form $\exists$ `Owner_r(C_Car_i)` extending the `Person` context. In the same way, 4 new `Car` concepts have been added in the `Car` lattice. For the `Garage` context, the number of concepts has not changed, but some concept intents have been modified to take into account the new concepts of the `Car` lattice.

[3] The $\bot$ concept is not considered when counting concepts in the following.

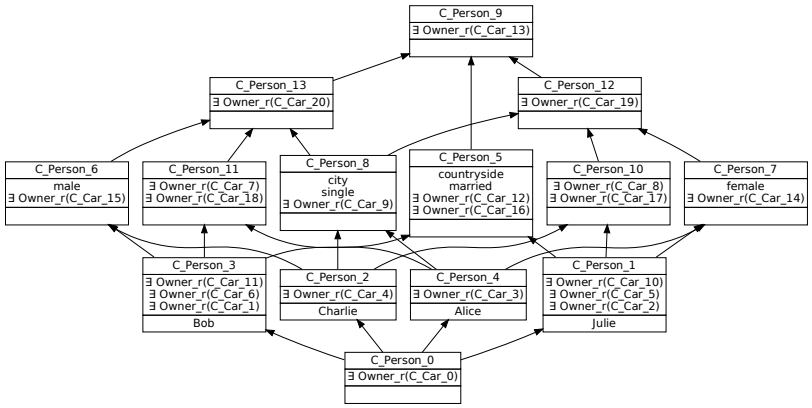

Fig. 7. `Person` lattice from RCA(RCF_1_r).

RCA(RCF_1_r) ≡ GCA(GC_1). A comparison of the results from RCA(RCF_1_r) and GCA(GC_1) shows that both are equivalent in term of concept extents. For illustration, let us look at the `Person` concepts in both approaches shown in the lattices in Fig. 7 for RCA and in Fig. 5 for GCA. Both lattices have the same number of concepts and concepts have the same extents. This example shows that when applying RCA to RCF_1 with inverse relations, the result RCA(RCF_1_r) is equivalent to the results obtained with GCA on the corresponding graph context GC_1. This example suggests that the addition of inverse relations leads to equivalent results for the two approaches. However, in the following, we show a case where the two approaches give different results, despite the addition of inverse relations.

RCA(RCF_2_r) ≢ GCA(GC_2). The example we are dealing with here is obtained by a small modification of the data presented in Fig. 3, `car6` is here maintained by `garage D` instead of `garage A`. The resulting graph context is called GC_2 and the corresponding relational context family with inverse relationships is named RCF_2_r. GCA produces one additional concept than RCA for this example (53 concepts against 52). Figure 8 shows the PGP `Q2x = ((x), P2x)` corresponding to this g-concept. In this PGP, the concept `Q2x` with `extent` = {`car5, car2, car1`} is the g-concept not constructed by RCA. Pattern `P2x` is the sub-graph containing the node `x`, the core nodes (in red), and all the other nodes useful for describing `x` identified in brackets, hence, the notation `x (bb l q w j ba)`.

We now describe what characterizes this additional g-concept. As shown in Fig. 8, the nodes adjacent to `x` are nodes `l, bb` and `w` defined by the relations: `Sell(l, x)`, `Maintain(bb, x)` and `Owner(x, w)`. The other nodes are introduced by these adjacent nodes and so on up to the core nodes that carry the main information of the pattern. A simplified description of `x` can be formulated as follows: `x` is the concept of cars sold by some garage `l`, maintained by some

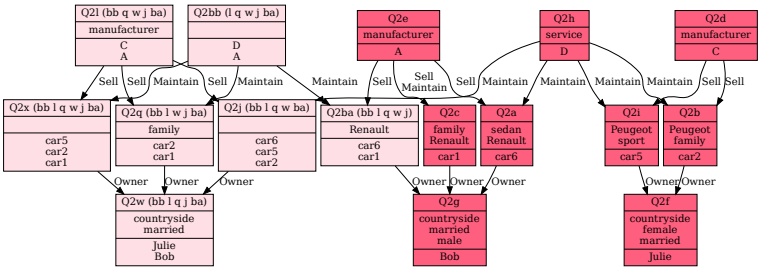

Fig. 8. `Q2x` PGP with `x` as projection tuple.

garage `bb` and owned by some person `w`, who also owns a family car `q` and another car `j`, both sold by the same garage `l`. This description includes the cycle `(l, q, w, j, l)`. At this point we suspect that this cycle may be the reason why RCA does not construct this concept. Note that both approaches produce the super-concept of `Q2x` whose extent is {`car6, car5, car2, car1`}.

In addition to this modelling case, we have also modified GC_1 in Fig. 3 to obtain four other data models by manipulating the different relationships between objects. For these data models, the results show the 2 previous cases: an equivalence of results (on 2 models) and the case where GCA produces more concepts than RCA (on 2 other models). In the latter case, we obtain for each model that GCA produces 2 g-concepts not found in RCA. The common point between these g-concepts is the presence of cycles in their associated pattern which may explain the fact that RCA does not construct the equivalent concepts. In the following, we further explore the effect of cycles in the data.

4.3 RCA and Graph-FCA Cycle Processing

We have described how despite the addition of inverse relations that brings the RCA results closer to those of GCA, there are situations where some g-concepts have no corresponding relational concepts. Analyses reveal that a common point of these g-concepts is the presence of cycles in their intent. Below, we study the effect of cycles in RCA and GCA results.

Cycles of length 2-4. The first example is a graph context (GC_3, Fig. 9) made of two cycles of length 2 and 4 formed on elements of the same category (`Person`) and defined on the relation `love(Person, Person)`. Figure 12 presents the GCA patterns – `Q1` (blue), `Q2` (red), `Q3` (lime-green), `Q4` (green) – and Fig. 14 presents the lattice view of these patterns.

As these patterns show, different nodes of a pattern may represent the same concept, e.g. nodes `a` and `b` in `Q2`. Indeed, they have equivalent intents and the same extent, pattern `Q2` describes two persons that love each other (a cycle of length 2). This duplication of concepts in GCA output is required to properly

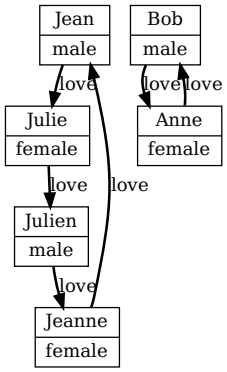

Fig. 9. Graph context made of cycles of length 2-4 (GC_3).

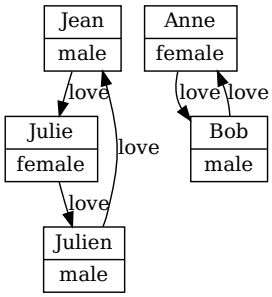

Fig. 10. Graph context made of cycles of length 2-3 (GC_4).

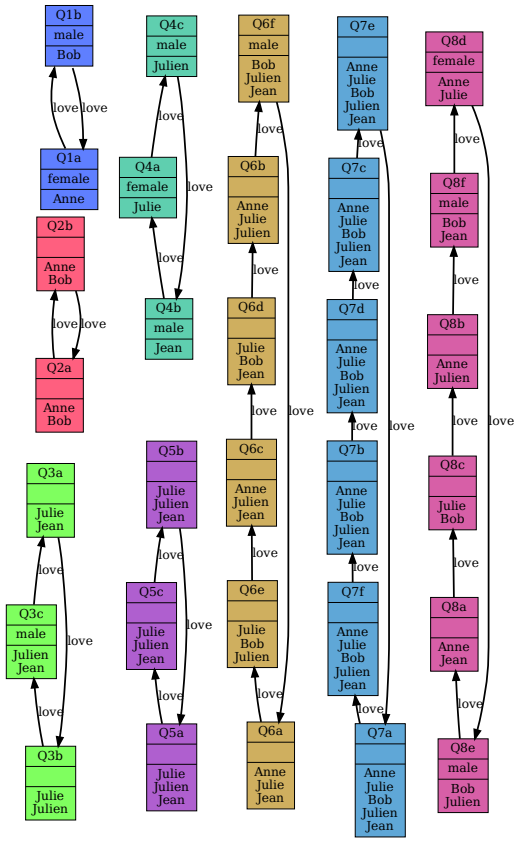

Fig. 11. GCA patterns for GC_4.

represent the graph patterns that exhibit symmetries. We refer to duplicated concepts as *automorphic concepts*, they are only different representations of the same theoretical concept. These automorphic concepts are represented in the hierarchical view by a meta-node grouping them (dashed box), for example `Q2a` and `Q2b` in Fig. 14. The top concept `Q3a-d` in Fig. 14, which includes all persons in its extent, is duplicated into 4 nodes organized into a cycle of length 4 (see Fig. 9). This particular pattern deserves an explanation. Looking at the data, it appears that all persons are involved in a cycle of length 2 or a cycle of length 4. The most specific generalization is therefore a cycle of length 4 because a cycle of length 2 can simulate a cycle of length 4 by going around the cycle twice.

Figure 13 presents the RCA corresponding lattice (in this case, the family lattice has only one lattice). The $\top$ concept `C_person_3` has $\exists$ `love(C_person_3)` as relational attribute. It is a self-referenced concept materializing the fact that *a person loves another person*. `C_person_3` is equivalent to the automorphic con-

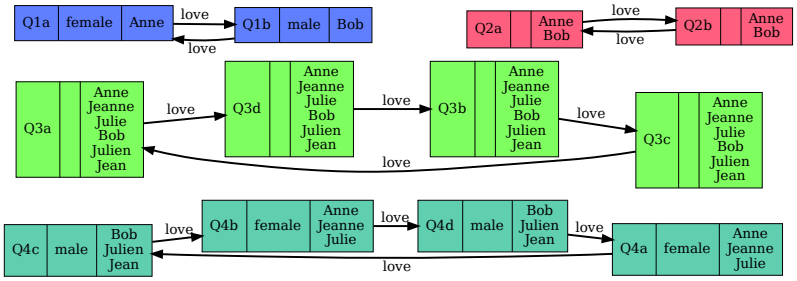

Fig. 12. GCA patterns for GC_3.

cepts that make up pattern `Q3` (they have the same extent). Nevertheless, as said before, their intents are defined differently and thus represent different information: `Q3` intent integrates the information on the cycle length through the automorphic concepts, while `C_person_3` intent represent a more abstract information. Concepts `C_person_1` and `C_person_2` pointing to each other through relational attributes represent the fact that *a male person loves a female person* and vice versa. `C_person_1` and `C_person_2` have the same extents as concepts `Q4a-b` and `Q4c-d` respectively, but not the same intents. In fact, like `Q3`, `Q4` is a cycle of length 4 and is made of two pairs of automorphic concepts as shown in Fig. 14.

Finally, taking into account the fact that automorphic concepts are considered identical, the concept lattice presented in Fig. 14 contains 6 concepts for GCA (`Q1a, Q1b, Q2a-b, Q4a-b, Q4c-d, Q3a-d`) against 3 concepts for RCA (`C_person_1, C_person_2, C_person_3`) as presented in Fig. 13. GCA concepts `Q1a, Q1b, Q2a-b` have no equivalent concepts in RCA, this could be explained by the fact that cycle lengths are not taken into account by RCA. For illustration, pattern `Q1` describes the fact that *a female and male persons love each other* (a cycle of length 2). In RCA, the two instances `Anne` and `Bob` have the same relational attributes as respectively `Julie`, `Julien`, i.e. $\exists love(male)$, $\exists love\_r(male)$, and $\exists love(female)$, $\exists love\_r(female)$. Thus all instances are grouped within the two concepts `C_person_1, C_person_2`.

Cycles of length 2-3. In this example, we modify GC_3 to see what happens when data contains cycles that do not have a multiple/divisor relationship between their lengths. This modification involves deleting node `Jeanne`, shortening the cycle of length 4 to length 3. This new graph context named GC_4 is shown in Fig. 10. The patterns produced by GCA for this example are presented in Fig. 11 and their lengths vary between 2, 3 and 6. Patterns of lengths 2 and 3 capture and generalize the structure of cycles in the data while those of length 6 generalize the cycles of lengths 2 and 3. For illustration, `Q1` and `Q4` form the graph context (GC_4), `Q2` generalizes `Q1`, `Q3` generalizes `Q4`, `Q5` generalizes `Q3` and `Q4` and so on. This reveals that in GCA patterns, the most specific generalization is a cycle whose length is the smallest common multiple of the cycle lengths in

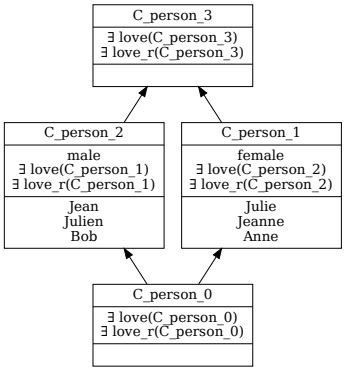

Fig. 13. RCA result about GC_3.

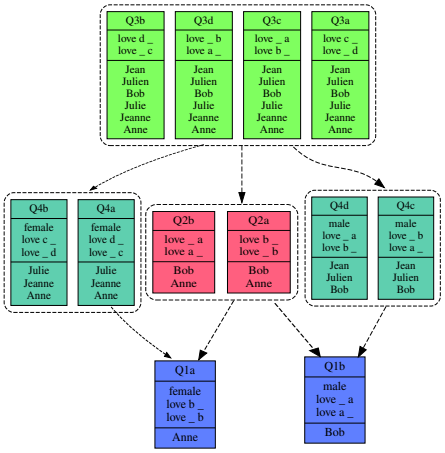

Fig. 14. GCA concept lattice about GC_3.

the graph context, just like in GC_3 where the most specific generalization was a cycle of length 4 which is the smallest common multiple of 2 and 4.

Conversely to the GC_3 example where the concept with extent {`Anne`} (`Q1a` in Fig. 14) had no equivalent in RCA results (Fig. 13), it is well found in the RCA results of GC_4 (concept `C_person_11` in Fig. 15). This is because, during the iterative process, relational attributes of {`Anne`} / {`Bob`} become different (pointing to different concepts) from these of other instances. RCA captures the difference between the two cycles – a female loves a male which loves a female / a female loves a male which loves a male – but not their lengths. GCA produces two additional g-concepts than RCA for GC_4 (23 concepts against 21): these automorphic concepts (`Q2a-b` and `Q5a-c` in Fig. 11) respectively characterize people involved in cycles of length 2 and 3, independently of their gender. In RCA, these two patterns are generalized in the self-referential top concept, that can be interpreted as *a person who loves a person, and is loved by a person.*

These analyses allow to conclude that, along with cycles in the data, some GCA graph patterns can contain or represent cycles that cannot be expressed in RCA, because cycles in RCA are taken into account through relational attributes pointing to concepts. Consequently, GCA concepts not built by RCA are due to the fact that GCA graph patterns capture some data structures that are covered in RCA by more general concepts.

4.4 Discussion

The examples and results described above have shown some similarities and differences between RCA and Graph-FCA, when applied on a common basis. Data are described with only unary and binary relations, the existential scaling quantifier is used for RCA, and only unary concepts are computed by GCA.

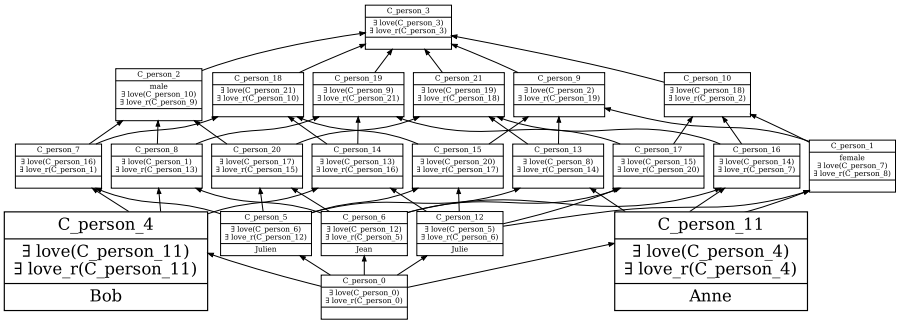

Fig. 15. RCA corresponding result on GC_4 where concepts 4 and 11 are highlighted.

Furthermore, concepts built by RCA and those built by GCA are compared on their extent. Finally, to make results closer, inverse relations have been included into RCA input data (RCF).

In general, the addition of inverse relations leads to an equivalence between the results of RCA and GCA, i.e. both methods produce the same number of concepts with the same extents. However, in some cases, GCA produces concepts not constructed by RCA. These additional concepts are due to the fact that intents in GCA are PGPs and the construction of graph patterns captures complex data structure, which can sometimes generate concepts that have no equivalence in RCA. The same reasons explain why data with cycles of different lengths are represented within several (possibly automorphic) concepts by GCA while RCA produces a few (possibly self-referencing) general concepts.

From an analyst point of view, the two methods have different qualities. On one hand, GCA has an ability to capture complex data structures, and to give a global view on data. On the other hand, RCA allows to progressively explore the data, choosing the relations to add and the scaling quantifiers to use.

5 Conclusion and Future Work

We have conducted a comparison between RCA and Graph-FCA within a restricted common framework, i.e., using existential scaling quantifier for RCA, unary concepts, data with only unary and binary relations for GCA and including inverse relations. Our study, based on several examples, highlighted the effect of data cycles on the results of the two methods. This result has to be explored further in order to dissect the underlying mechanisms in the two methods.

In the future, other points of comparison will be explored between RCA and GCA. Our work will focus on the study of n-ary concepts implemented in GCA and how RCA unary concepts can be used to construct n-ary concepts that come close to the GCA intuition. It would also be interesting to compare both approaches in less similar configurations: encoding n-ary relations by reification

and using other quantifiers in RCA. Other effects of intent definitions and the different parameters handled by these approaches can also be studied.

References

1. Alam, M., Coulet, A., Napoli, A., Smaïl-Tabbone, M.: Formal concept analysis applied to transcriptomic data. In: FCA4AI (ECAI 2012) (2012)
2. Braud, A., Dolques, X., Huchard, M., Le Ber, F.: Generalization effect of quantifiers in a classification based on relational concept analysis. Knowledge-based systems **160**, 119–135 (2018)
3. Ferré, S.: A proposal for extending formal concept analysis to knowledge graphs. In: Formal Concept Analysis: ICFCA 2015, Proc. pp. 271–286. Springer (2015)
4. Ferré, S., Cellier, P.: How Hierarchies of Concept Graphs Can Facilitate the Interpretation of RCA Lattices? In: CLA 2018. CEUR-WS Proc., vol. 2123 (2018)
5. Ferré, S., Cellier, P.: Graph-FCA: An extension of formal concept analysis to knowledge graphs. Discrete applied mathematics **273**, 81–102 (2020)
6. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1999)
7. Hahn, G., Tardif, C.: Graph homomorphisms: structure and symmetry. In: Graph symmetry: algebraic methods and applications, pp. 107–166. Springer (1997)
8. Hitzler, P., Krotzsch, M., Rudolph, S.: Foundations of semantic web technologies. Chapman and Hall/CRC (2009)
9. Keip, P., Ferré, S., Gutierrez, A., Huchard, M., Silvie, P., Martin, P.: Practical comparison of fca extensions to model indeterminate value of ternary data. In: CLA 2020. CEUR-WS Proc., vol. 2668, pp. 197–208 (2020)
10. Keip, P., Gutierrez, A., Huchard, M., Le Ber, F., Sarter, S., Silvie, P., Martin, P.: Effects of Input Data Formalisation in Relational Concept Analysis for a Data Model with a Ternary Relation. In: ICFCA 2019 - 15th Int. Conf. on Formal Concept Analysis. LNCS, vol. 11511, pp. 191–207 (2019)
11. Kötters, J.: Concept lattices of a relational structure. In: Conceptual Structures for STEM Research and Education: ICCS 2013, Proc. pp. 301–310. Springer (2013)
12. Lehmann, F., Wille, R.: A triadic approach to formal concept analysis. In: Conceptual Structures: Applications, Implementation and Theory: ICCS'95, Proc. pp. 32–43. Springer (1995)
13. Nica, C., Braud, A., Dolques, X., Huchard, M., Le Ber, F.: Extracting hierarchies of closed partially-ordered patterns using relational concept analysis. In: Graph-Based Representation and Reasoning. ICCS 2016. LNCS vol 9717, Springer (2016)
14. Nica, C., Braud, A., Le Ber, F.: RCA-SEQ: an original approach for enhancing the analysis of sequential data based on hierarchies of multilevel closed partially-ordered patterns. Discrete Applied Mathematics **273**, 232–251 (2020)
15. Priss, U.: Formal concept analysis in information science. Annu. Rev. Inf. Sci. Technol. **40**(1), 521–543 (2006)
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**, 81–108 (2013)
17. Sowa, J.F.: Conceptual structures: information processing in mind and machine. Addison-Wesley Longman Publishing Co., Inc. (1984)
18. Wille, R.: Conceptual graphs and formal concept analysis. In: Conceptual Structures: Fulfilling Peirce's Dream: ICCS'97, Proc. pp. 290–303. Springer (1997)