Relational Concept Analysis in Practice: Capitalizing on Data Modeling using Design Patterns

Agnès Braud, Xavier Dolques, Marianne Huchard, Florence Le Ber, Pierre Martin

Relational Concept Analysis in Practice: Capitalizing on Data Modeling using Design Patterns

Agnès Braud[1][0000−0003−3614−9141], Xavier Dolques[1][0000−0002−5579−1714], Marianne Huchard[2][0000−0002−6309−7503], Florence Le Ber[1][0000−0002−2415−7606], and Pierre Martin[3][0000−0002−4874−5795]

[1] Université de Strasbourg, CNRS, ENGEES, ICube UMR 7537, F-67000 Strasbourg, France
firstname.lastname@unistra.fr, florence.leber@engees.unistra.fr
[2] LIRMM, Univ. Montpellier, CNRS, Montpellier, France
marianne.huchard@lirmm
[3] CIRAD, UPR AIDA, F-34398 Montpellier, France
pierre.martin@cirad.fr

Abstract. Many applications of Formal Concept Analysis (FCA) and its diverse extensions have been carried out in recent years. Among these extensions, Relational Concept Analysis (RCA) is one approach for addressing knowledge discovery in multi-relational datasets. Applying RCA requires stating a question of interest and encoding the dataset into the input RCA data model, i.e. an Entity-Relationship model with only Boolean attributes in the entity description and unidirectional binary relationships. From the various concrete RCA applications, recurring encoding patterns can be observed, that we aim to capitalize taking software engineering design patterns as a source of inspiration. This capitalization work intends to rationalize and facilitate encoding in future RCA applications. In this paper, we describe an approach for defining such design patterns, and we present two design patterns: "Separate/Gather Views" and "Level Relations".

Keywords: Formal Concept Analysis · Relational Concept Analysis · Design patterns.

1 Introduction

Formal Concept Analysis (FCA [15]) has gained importance in knowledge discovery thanks to both theoretical advances and the multiplication of concrete application projects in many domains [31]. Part of these advances are extensions of FCA suitable to handle complex data, going far beyond basic formal contexts. A few of these extensions are dedicated to deal with datasets comprising multiple object categories and multiple relationships between these objects, namely Graph-FCA [12], Relational Concept Analysis (RCA) [17], and the approach described by [22]. Each of these approaches has its own practical application

and implementation constraints, but has specific qualities for knowledge discovery. E.g. Graph-FCA provides concepts highlighting graph patterns shared by tuples, while RCA provides interconnected concept lattices, one per object category. This paper focuses on RCA.

RCA is based on a simple input data model composed of objects described by Boolean attributes and unidirectional binary relationships between these objects. Practical application raised the issue of encoding a dataset into this formalism, which was more or less easy according to the dataset model structure, such as converting a ternary relation into binary relations [21]. Similar problems arise for encoding object descriptions with particular attribute values (e.g. numerical) into a formal context made of Boolean ones. The latter can be addressed, for example, using scaling approaches [15] or Pattern Structures [14]. To facilitate access of new users to RCA, capitalizing the experience gained in applying RCA in various existing applications is a need.

This paper aims to describe a general approach, to pave the way for the definition of design patterns for RCA application. To this end, we present what such design patterns might look like, and give a few illustrations. Section 2 presents basics of RCA, some typical applications, and our motivation for capitalizing the encoding practices as design patterns. Section 3 outlines the design pattern notion, inspired by its definition in the field of software engineering, and illustrates it through two examples. Section 4 discusses opportunities for developing the approach. We conclude and give a few perspectives of this work in Sect. 5.

2 Background and motivation

Formal Concept Analysis FCA is a mathematical framework focusing on the analysis of binary relations using concept lattices. FCA considers as input a formal context $\mathcal{K} = (O, A, I)$ where O is a set of objects, A is a set of attributes and $I \subseteq O \times A$ is the incidence relation. We define the functions $f : \mathcal{P}(O) \rightarrow \mathcal{P}(A)$ and $g : \mathcal{P}(A) \rightarrow \mathcal{P}(O)$ such that $f(X) = \{y \in A | X \times \{y\} \subseteq I\}$ and $g(Y) = \{x \in O | \{x\} \times Y \subseteq I\}$. The concept lattice $\mathcal{L}$ computed from $\mathcal{K}$ is the set of concepts $\{(X, Y) | X \subseteq O, Y \subseteq A, f(X) = Y \text{and } g(Y) = X\}$, provided with a partial order relation based on inclusion. X is the concept extent, Y is the concept intent. A concept (X_{sub}, Y_{sub}) is lower in the lattice, i.e. is a sub-concept of a concept (X_{sup}, Y_{sup}) when $X_{sub} \subseteq X_{sup}$.

Relational Concept Analysis RCA aims at extending FCA to take into account a dataset where objects of several categories are described by attributes and by relations to objects [17]. The dataset is called a Relational Context Family (RCF). An RCF is a $(\mathbf{K}, \mathbf{R})$ pair where: $\mathbf{K} = \{\mathcal{K}_i\}_{i=1,\ldots,n}$ is a set of $\mathcal{K}_i = (O_i, A_i, I_i)$ contexts (i.e. Formal Contexts), and $\mathbf{R} = \{r_j\}_{j=1,\ldots,m}$ is a set of r_j relations (i.e. Relational Contexts) where $r_j \subseteq O_{i_1} \times O_{i_2}$ for some $i_1, i_2 \in \{1, \ldots, n\}$. An example of an RCF composed of two formal contexts introducing pests and crops respectively and one relational context indicating which pest attacks which crop is shown in Table 1. From the RCF, RCA iteratively builds concepts. In

FC Pest	diptera	beetle	hemiptera	insect	fungi
Contarinia tritici	x			x	
Oulema melanopa		x		x	
Lepidosaphes ulmi			x	x	
Verticillium dalhiae					x

FC Crop	cereal	fruit tree
wheat	x	
barley	x	
apple tree		x
apricot tree		x

RC attacks	wheat	barley	apple tree	apricot tree
Contarinia tritici	x			
Oulema melanopa		x		
Lepidosaphes ulmi			x	
Verticillium dalhiae				x

Table 1. Example of a Relational Context Family made of the Formal Contexts `Pest` and `Crop`, and the Relational Context `attacks`.

a first step, concepts are built for each formal context, e.g. concept `C_Crop_15` that groups `wheat` and `barley` for the common attribute `cereal` (right-hand side lattice of Fig. 1). At the next step, the relational context `attacks` is used to form *relational attributes* that express a relation that a pest object may have with a crop concept, such as ∃`attacks(C_Crop_15)` assigned to `Contarinia triciti` and `Oulema melanopa` because *they attack at least one crop of* `C_Crop_15`. This causes the creation of concept `C_Pest_18`, which would not be there without ∃`attacks(C_Crop_15)` (left-hand side lattice of Fig. 1). The relational attributes can be formed with different quantifiers (e.g. ∃∀, ⊇, or with percentages). The number of iterations depends on the data model. The same process can be applied to more complex datasets, eventually containing circuits.

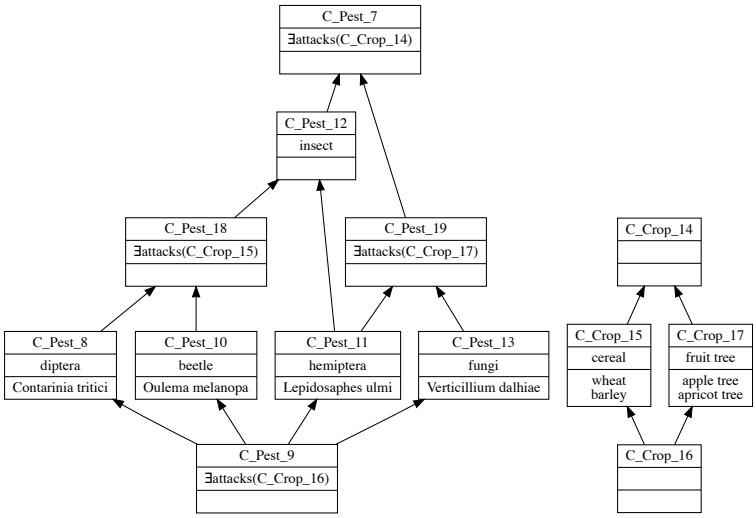

Fig. 1. Lattices obtained from the Relational Context Family of Table 1.

Overview on RCA applications RCA was used for analyzing data in various domains. Hydroecological datasets were studied in [11,28], e.g. to correlate physico-

chemical parameters and characteristics of taxons living in sample sites. Agro-ecology, and more precisely finding plant-based extracts that can serve as alternative to synthetic pesticides and antimicrobials, was considered in [24]. Using RCA in Information Retrieval (IR) is discussed in [8], and was, for instance, used for querying collections of legal documents connected through cross references [25]. In the field of Semantic Web, it was for instance used to design ontologies [32,20] and extract consistent families of dependent link key candidates in RDF data [4]. RCA was used in industrial information systems to make tools inter-operate, e.g. in the domain of brain activity measurement [35]. It was also applied to identify anomalies in aluminum die casting process description [34]. Finally, RCA was applied in Software Engineering in order to solve different problems: normalize (by factorization) a UML class model [19,16] or a UML use case model [10]; for refactoring [26]; learn model transformation patterns [9]; build directories of Web services or components [5]; structure [7] or analyze [2,18] the variability in software product lines.

Motivation Applying RCA to a dataset is easy when their data models are compliant. But the model of the dataset is often more complex, and has to be transformed. For instance, it may contain bidirectional or N-ary relations. The relations may have specific semantics, e.g. *is-a*, *instance-of*, and *contains*. In such case, converting an *is-a* relation connecting an entity E_{sub} to another entity E_{super} into an RCA relation conducts to assign attributes of E_{super} to both entities. Identical problems arise when applying FCA to multi-valued attributes. A solution is to use one of the *scaling* schemes [15] that convert non-Boolean attributes into Boolean ones. Mastering these *scaling* schemes is a key for FCA practitioners to select the most appropriate one to meet the modeling objective. The aim of this work is therefore to capitalize on the recurrent data encoding patterns for RCA and thus to increase the efficiency of RCA practitioners in developing applications. In this work, the considered encoding includes values, instances (objects), and data model transformations. The main source of inspiration for this work originated from the domain of Software Engineering, namely the design patterns [13].

3 Design Patterns

This section introduces our proposed description of design patterns (DPs) for RCA (Section 3.1) and presents two DPs: *Separate/Gather Views* (Section 3.2) and *Level Relations* (Section 3.3). The first DP aims to handle (i.e. separates or groups) attributes to facilitate the analysis of a dataset through specific views, and the second one to represent multivalued attributes from the same category discretized into the same set of values using both relational and formal contexts. Both DPs were used in different applications.

3.1 Describing a Design Pattern

Data modeling is a recurrent and fundamental problem in various approaches or domains, including databases, ontologies or software engineering. This last domain is the one where capitalization and sharing of models have been the most developed. The work by Gamma *et al.* [13] is a seminal reference for DP in the domain of Software Engineering. It has been inspired by the work of the architect Christophe Alexander [3]. Gamma *et al.* offers a catalog of DPs for object-oriented designers to transfer knowledge, in particular from experts to beginners, to help them achieve a good design by choosing among several proven alternatives. In the catalog, four essential parts are highlighted to describe a DP: the pattern name, the problem, the solution, and the consequences. The *pattern name* is important as it becomes part of a shared vocabulary. Describing the *problem* involves defining the context and the favorable conditions of the DP application. The *solution* describes the different elements that take part to the design in an abstract and configurable way. The *consequences* include predictable effects of the pattern application and compromises that may have to be done between different qualities of the results. In addition, each DP is described in the catalog using more than ten sections. While some sections are specific to object-oriented design (e.g. classes/objects participants, or sample code), others can be adopted for other domains. In this paper, to remain synthetic, the following five description sections are used:

- **Problem**. The problem is expressed in terms of dataset content and analysis objective. As in [13], this is the most tricky part of the description.
- **Solution**. For RCA, the solution consists in expressing how to formally design and populate a Relational Context Family from the problem description.
- **Example**. This section presents a short example of the pattern application.
- **Known uses**. This section reports existing case studies of the literature where the DP has been applied.
- **Consequences**. This section reviews alternatives and discusses the consequences of the application of the DP relatively to the analysis objective, in particular in terms of usability/readability of the result.

3.2 The Design Pattern *Separate/Gather Views*

Problem This design pattern applies when:

- The objects identified in the dataset are described either by attributes with domain values of various cardinalities (Case 1), or by Boolean attributes (Case 2). Case 1 can be reduced to Case 2 using a scaling operation [15]. Attributes can be gathered into groups and categories for Case 1 and 2 respectively;
- Each attribute or attribute group of Case 1 or each Boolean attribute category of Case 2 is a coherent *view* on objects;
- It is relevant to analyze objects through the perspective of *a single view* or considering *several views*, first separately and then together.

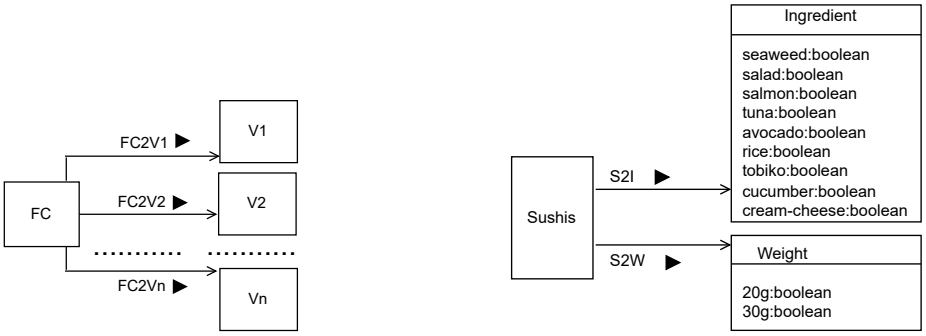

Fig. 2. Schema of the Relational Context Family for the Design Pattern *Separate/Gather Views* (left-hand side). Schema for the sushis example (right-hand side). Both schemas are represented with the UML class diagram notation.

Solution The solution, outlined in Figure 2 (left-hand side), is defined as follows:

- Formal contexts
  - One formal context denoted as FC for the initial objects
    - FC objects (O) are the initial objects, FC attributes (A) are initial object identifiers, or other description, or none: $FC = (O, A, R)$, $R \subseteq O \times A$
  - One formal context denoted as V_i for each view $i, 1 \leq i \leq n$:
    - V_i objects (O_i) are views on the initial objects, V_i attributes (A_i) are Boolean attributes of one view: $V_i = (O_i, A_i, R_i)$, $R_i \subseteq O_i \times A_i$. For each view i, there is a one-to-one mapping between objects of O and their projection in O_i denoted as $proj_i : O \longrightarrow O_i$.
- Relational contexts
  - For each $i, 1 \leq i \leq n$, one relational context $FC2V_i$ connects an object of FC to its corresponding view object in a V_i: $FC2V_i = (O, O_i, r_i)$, $r_i \subseteq O \times O_i$, $r_i = \{(o, proj_i(o)) | o \in O\}$

Example Table 2 presents the dataset, i.e. sushis described by weight and ingredients, with domain values $\{20g, 30g\}$ and $\{seaweed, ...cream\_cheese\}$ respectively. Applying the DP *Separate/Gather Views* to this dataset consists in identifying the objects and the views. As Sushis are the objects, weight and ingredients are candidates for the views. Figure 2 (right-hand side) graphically outlines the DP application with a UML class model. Class `Sushis` represents the main formal context; classes `Ingredient` and `Weight` represent secondary formal contexts (the views on sushis). UML associations `S2I` and `S2W` represent the relational contexts connecting each object to its view object. The values of the attributes `Weight` and `Ingredient` lead to Boolean attributes such as `seaweed` (in `Ingredients`) or `20g` (in `Weight`). Table 3 shows the resulting RCF: FC is Table `Sushis`, where there are only objects and no additional description by attributes. V_1 is Table `Weight`, which associates weight views of sushis (e.g.

`california salmon Wght`) with the corresponding weight (e.g. `20g`). V_2 is Table `Ingredient`, which associates ingredient views of sushis (e.g. `california salmon Ing`) with the corresponding ingredients (e.g. `seaweed`). $RC_1 = S2W$ connects a sushi to its weight view; $RC_2 = S2I$ connects a sushi to its ingredient view.

Sushis	Weight	Ingredients
california salmon	20g	seaweed, salad, salmon, avocado, rice
california	30g	seaweed, salad, tuna, avocado, rice
maki cheese	20g	salad, avocado, rice, cream-cheese
maki tobiko	30g	seaweed, avocado, rice, tobiko, cucumber

Table 2. A tiny sushi dataset. A sushi is described by its weight and its ingredients.

FC Sushis
california salmon
california tuna
maki cheese
maki tobiko

V1 Weight	20g	30g
california salmon Wght	x	
california tuna Wght		x
maki cheese Wght	x	
maki tobiko Wght		x

V2 Ingredient	seaweed	salad	salmon	tuna	avocado	rice	tobiko	cucumber	cream-cheese
california salmon Ing	x	x	x		x	x			
california tuna Ing	x	x		x	x	x			
maki cheese Ing		x			x	x			x
maki tobiko Ing	x				x	x	x	x	

RC1 S2W	california salmon Wght	california tuna Wght	maki cheese Wght	maki tobiko Wght
california salmon	x			
california tuna		x		
maki cheese			x	
maki tobiko				x

RC2 S2I	california salmon Ing	california tuna Ing	maki cheese Ing	maki tobiko Ing
california salmon	x			
california tuna		x		
maki cheese			x	
maki tobiko				x

Table 3. A Relational Concept Family for sushis: FC Sushis, V_1 weight view, V_2 ingredient view, $RC_1 = S2W$ connects a sushi to its weight view, $RC_2 = S2I$ connects a sushi to its ingredient view.

Using this encoding, RCA builds the concept lattices presented in Fig. 3 and 4. Figure 3 presents the views for weight and for ingredients. The weight concept lattice helps analyzing groups of sushis sharing the same weight, i.e. 20g versus 30g. The ingredient concept lattice highlights other sushi groups, e.g. `Concept_Ingredient_11` groups the California sushis (separated in the weight view) because of their four shared ingredients (`avocado, rice, seaweed, salad`). Figure 4 gathers both views and gives combined information, such as the implication rule that a sushi weighed 20g contains salad:
$\exists S2W(\texttt{Concept\_Weight\_19}) \longrightarrow \exists S2I(\texttt{Concept\_Ingredient\_12})$.

In this example, the resulting RCA lattice is similar to a FCA lattice. But each view could be more complex, e.g. with hierarchical values or a combination

of various types of attributes with consistent semantics. This approach justifies to build a lattice for each view and one lattice gathering all views.

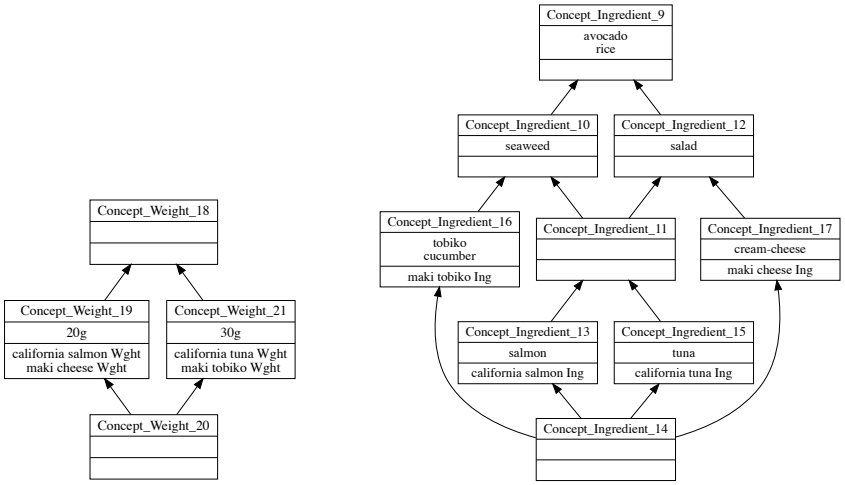

Fig. 3. Views on sushis: according to their weight (left-hand side) and their ingredients (right-hand side).

Known Uses This DP has been implemented for the analysis of visual accessibility options in operating systems (OS) in [23]. This analysis had different purposes, including making recommendations to OS developers to design a new version, to assist end-users finding an accessibility configuration close to the current configuration when the OS upgrades, or when end-users have to change OS. In this study, the objects are operating systems (OS) and the views cover three visual accessibility options categories (contrast, text, zoom). Separating the views allows to analyze the OS along a single problematic, e.g. to observe commonly shared contrast options, which OS provides more contrast options than another, or which options are never provided together. Gathering the views classifies the OS in a global way, e.g. helping identifying which ones are equivalent on all option categories, how they differ from each other, and how the options of the different categories interact. Moreover, an application has been developed by [18] using this DP to assist *Feature location (FL)* in Software Product Line Engineering.

Consequences Part of the value of this pattern relies on the relevance of the designed views. It may be more or less complex to determine which set of attributes corresponds to a view, as a single semantics should be associated with this view,

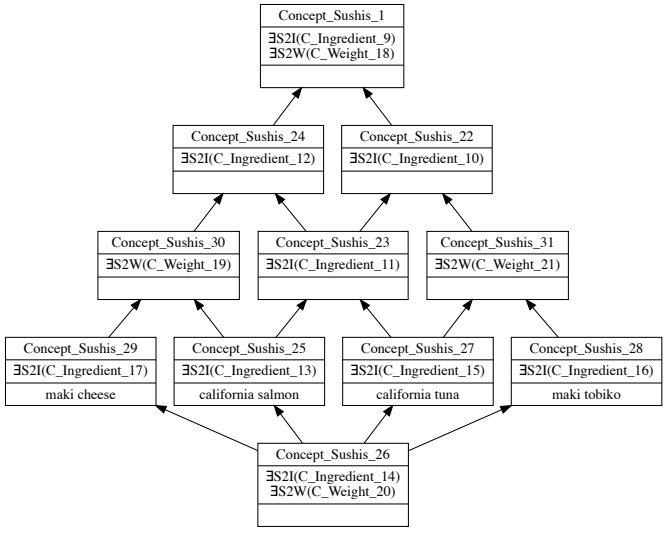

Fig. 4. Sushis concept lattice (gathered views).

e.g. *habitat* versus *food* for animals. Note that partitioning attributes participating to different coherent sets is not relevant, e.g. for animals, the Boolean attribute *aquatic environment* can be an attribute in views *natural habitat* and *growing conditions*. An additional interest of using this RCA DP may occur when considering the pattern variation in which an object can have several views, corresponding for instance to different versions. Different quantifiers can thus be used in the diverse relations, e.g. in the Sushis example, $\exists$ on $S2W$ and $\exists\forall$ on $S2I$, to find sushis having versions containing only certain types of ingredients, such as vegan ingredients, considering that this ingredient description was included in V_2 `Ingredient`.

One might wonder whether using FCA on (1) each formal context V_i, and (2) on a simple formal context gathering all the Boolean attributes would be more relevant than applying this pattern, in particular to obtain more readable lattices since relational attributes add some reading complexity. Using the single formal context with all attributes would present these drawbacks: if some attributes of a view are equivalent and correspond to an interesting abstraction (shared by the same object set), then they would be mixed with other attributes of the other views where it will be difficult to observe the equivalence, the implication, or the mutual exclusion between different abstractions coming from different views. In addition, if the description of the attributes is complex, FCA cannot consider this complexity to classify the objects while, in the RCA framework, this complexity can be converted into objects supporting this description. It consists in extending the Relational Context Family in which the whole description is

used for classifying the initial objects. To improve the readability of the relational attributes, which is a drawback of RCA, the simplification proposed by [33,34] can be applied: a concept identifier in a relational attribute can be replaced by the intent of the concept at its creation time (possibly recursively). Other authors proposed to deliver the extracted knowledge, not in terms of concepts and relational attributes, but as sentences [29] or as logical formulas in source code macros [18].

3.3 The Design Pattern *Level Relations*

Problem This design pattern applies when:

- The objects identified in the dataset are described by numerical attributes. The attributes belong to the same semantic category;
- Numerical attributes are discretized into the same set of values;
- Value Levels give interesting information by themselves;
- It is relevant to factorize values to limit the size of the context. Different quantifiers can be applied to the various levels.

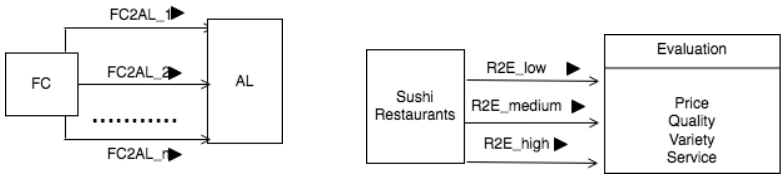

Fig. 5. Schema of the Relational Context Family for the Design Pattern *Level Relations* (left-hand-side). Schema for the sushi restaurant example (right-hand side).

Solution The solution is outlined in Fig. 5 (left-hand-side) and defined as follows:

- Formal contexts
  - One formal context denoted as FC for the initial objects:
    - FC objects (O) are the initial objects, FC attributes (A) are initial object identifiers, or other description, or none: FC $= (O, A, I)$, $I \subseteq O \times A$
  - One formal context denoted as AL:
    - AL objects (Ω) are the initial attributes, AL attributes (B) are attribute identifiers or categories of attributes: AL $= (\Omega, B, J)$, $J \subseteq \Omega \times B$.
- Relational contexts
  - For $i \in [1, n]$, n being the number of levels, the relational context FC2AL_i connects an object of FC to an attribute in AL if i is the level of this attribute for the object: FC2AL_$i = (O, \Omega, r_i)$, $r_i = \{(o, \omega) | o \in O, \omega \in \Omega, i$ is the value of $\omega(o)$ in the initial dataset$\}$

Example Table 4 outlines the dataset of the tiny example: sushi restaurants are described by their evaluation on price, food quality, food variety, and service, being multi-valued attributes with the three same values. Figure 5 (right-hand side) graphically outlines the DP application with an UML class model. Class `Sushi Restaurants` represents the main formal context; Class `Evaluation` represents a secondary formal context (the attributes). UML associations `R2E_Low`, `R2E_Medium` and `R2E_High` represent the relational contexts connecting each object to each attribute according to its value level.

Applying the DP *Level Relations* to Table 4 conducted to obtain Table 5. Two objects contexts are defined, i.e. one for the restaurants, the other for the evaluations. Three relational contexts are defined, i.e. one for each evaluation level (low, medium, and high). The lattices obtained from this RCF are shown on Fig.

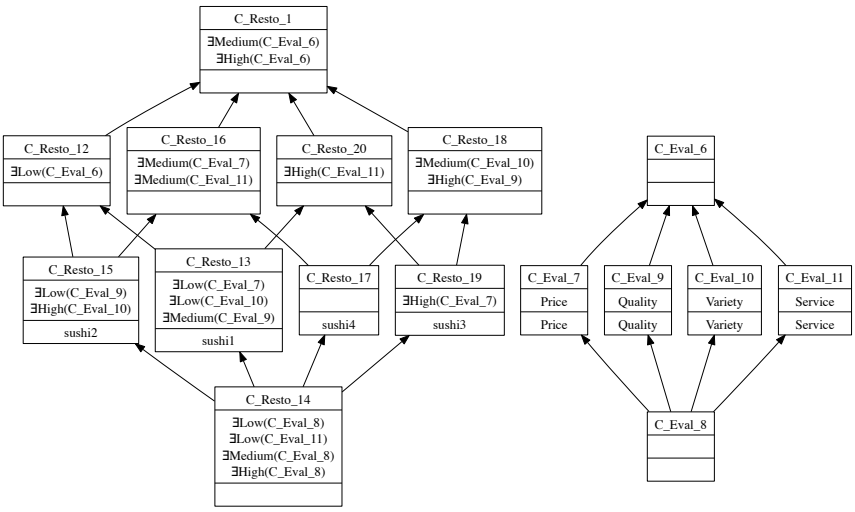

Fig. 6. Sushis restaurant concept lattices. For the sake of space: `Restaurant` has been shorten in `Resto`, `Evaluation` has been shorten in `Eval`, `R2E` has been removed before `Low`, `Medium` and `High`.

Restaurant	Price	Quality	Variety	Service
sushi1	low	medium	low	high
sushi2	medium	low	high	medium
sushi3	high	high	medium	high
sushi4	medium	high	medium	medium

Table 4. A tiny sushi restaurant evaluation.

Restaurants
sushi1
sushi2
sushi3
sushi4

Evaluation
Price
Quality
Variety
Service

R2E-Low	Price	Quality	Variety	Service
sushi1	x		x	
sushi2		x		
sushi3				
sushi2				

R2E-Medium	Price	Quality	Variety	Service
sushi1		x		
sushi2	x			x
sushi3			x	
sushi2	x		x	x

R2E-High	Price	Quality	Variety	Service
sushi1				x
sushi2			x	
sushi3	x	x		x
sushi2		x		

Table 5. A Relational Concept Family for sushis restaurants: *FC* Restaurant, *AL* Evaluation. Identifiers of objects are not shown for a sake of space. R_1=R2E-Low connects a sushi restaurant to its low evaluations, R_2=R2E-Medium connects a sushi restaurant to its medium evaluations, R_3=R2E-High connects a sushi restaurant to its high evaluations.

6. Concepts of the restaurant lattice (left on Fig. 6) have attributes pointing to evaluation concepts (right on Fig. 6). For instance, concept `C_Resto_16` gathers restaurants that have a medium evaluation on Price and on Service. Concept `C_Resto_1` gathers all restaurants, showing that all have both a medium and a high evaluation on some criteria.

Known Uses DP *Level Relations* has been applied for modeling a water dataset, where stream sites are described by physico-chemical parameters (pH, nitrates, phosphates, etc.) and fauna information [11]. Data are modeled with several relations to represent the different value levels of physico-chemical parameters (five level relations), macro-invertebrate populations (three level relations), and characteristics (life traits) of the macro-invertebrates (six level relations). Finally the RCF contains four formal contexts (stream sites, physico-chemical parameters, life traits and macro-invertebrates) and 14 relational contexts.

Moreover, this DP was also used by [27] to analyze a dataset on hydrosystem restoration projects. In this work, river sites are characterized by temporal heterogeneous information, including measures of physico-chemical parameters, biological indicators, and the composition of the surrounding land use. Biological indicators are described with 5 quality values. Physico-chemical parameters are discretized into 5 values. Land use is assessed within two increasing buffers. Part of different types of land use is summarized into three values, low, medium and high. The river sites belong to river segments where restoration operations have been undertaken. Finally the RCF contains 6 formal contexts and 21 relational contexts.

Consequences The lattice on sushi restaurants (left on Fig. 6, called lattice R in the following) obtained with RCA based on the DP *Level Relations* applied on the dataset described in Table 4 can be compared with a lattice obtained with a simple scaling of the multi-valued attributes into Boolean ones. This second lattice, called lattice B in the following, is shown in Fig. 7. Concepts introducing

the objects (`restaurant`) are similar in both lattices. More general concepts are different: lattice R has one more concept, i.e. `C_Resto_12`, which groups restaurants having a low evaluation on some criteria. In such case, RCA produces relational attributes in which only the level ($\exists$ *medium* / *high*) is represented and not the value (which attribute). Such attributes can be interesting for experts. Furthermore the tops of the lattices are different: it is empty in lattice B while it contains two relational attributes in lattice R. DP *Level Relations* is thus relevant when values provide an interesting information.

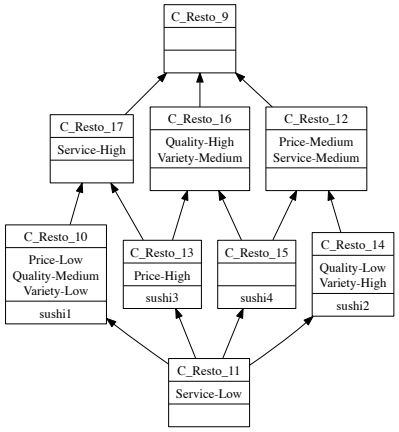

Fig. 7. Lattice obtained from Table 4 with a scaling of multi-valued attributes.

4 Perspectives

Section 3 presented two DPs, that were used in concrete applications, focusing on various aspects of relational data, as can be considered in the RCA framework.

Several perspectives are offered by this work. Two of them are discussed in this section, i.e. identifying additional DPs that can be useful to help RCA users and revisiting existing applications.

New opportunities for defining DPs Additional DPs were identified along the existing applications. A few examples are introduced hereafter. Some data models included a specialization/generalization (*is-a*) relationship. Analyzing the associated dataset required to flatten the hierarchical descriptions [19,10,16], to discover abstractions previously hidden. This new DP could be named **Collapse Specialization**. The DP **Reify Relation** may consist in introducing a formal context in the RCF for describing the tuples of a relation. This approach

was used by [21] to derive a binary representation of an N-ary relation, without loosing information. The link reification, as realized in [9] and [4], could be considered through the same perspective. The DP, dual to *Separate/Gather Views*, is **Separate/Gather Objects**, in which objects of a same category can be separated into several subsets to analyze each subset apart or all subsets as a whole. This approach was considered by [16] to normalize a UML class model to detect candidate attribute generalizations on diverse criteria, e.g. name, substring in the name, synonyms, type, default value. The DP **Instances2Model** may address objects with multi-valued attributes to extract a schema, as it has been done from instance descriptions in [9]. It consists in simplifying the description considering that an object has a value for an attribute (but not the value). Introducing virtual objects representing a **Query** has been proposed by authors in the context of FCA. This can be extended to the context of RCA, such as in [6], introducing several virtual objects in formal contexts and virtual links in relational contexts. This approach has been adopted for solving the problem of replacing a failing web service in a web service workflow [5]. Finally, sequences can be modeled into RCA input with a transitive relation 'precedes' or 'succeeds', according to the DP **Temporal** [28]. Other transitive relations (e.g. part-of/includes, or down/upstream-of [27]) may also correspond to this DP.

Revisiting applications with the DPs The DP *Separate/Gather Views* presented in Section 3 could be applied to revisit and improve a previous work on component catalog FCA-based building [1], in which software components were described with provided/required interfaces, each interface containing a set of services. In this catalog, the description was made using a single formal context. The catalog then organized the software components with this description and exposed possible substitution between components from the two viewpoints as a whole (provided as well as required services). This hardly helps the lattice exploration considering only one viewpoint, e.g. a user may want to search a component with provided services, and consider in a second step the required services, that other components could provide. Another potential use of the DP *Separate/Gather Views* could be to consider positive versus negative description of objects, a question that has been addressed in [30], to complete their approach using an additional point of view.

5 Conclusion

This paper presented a general approach that aims to capitalize lessons learned in encoding datasets in the RCA input format. Two DPs were described to illustrate the approach. These DPs have been extracted from existing RCA applications, and could be applied to improve or complete the analysis in other applications. Additional DPs to formalize were also reported. Short-term future work includes continuing the DPs formalization and defining a data modeling process involving the DPs. In the longer term, a new area of research could be extending this approach to other FCA extensions, such as Pattern Structures, Graph-FCA, and

Polyadic FCA. Using DPs could be a gateway to use several FCA extensions in synergy, as they could improve their implementation in case studies.

Acknowledgements This work was supported by the ANR SmartFCA project, Grant ANR-21-CE23-0023 of the French National Research Agency.

References

1. Aboud, N., Arévalo, G., Bendavid, O., Falleri, J., Haderer, N., Huchard, M., Tibermacine, C., Urtado, C., Vauttier, S.: Building hierarchical component directories. J. Object Technol. **18**(1), 2:1–37 (2019)
2. Al-Msie'deen, R., Seriai, A., Huchard, M., Urtado, C., Vauttier, S.: Documenting the mined feature implementations from the object-oriented source code of a collection of software product variants. In: 6th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE). pp. 138–143 (2014)
3. Alexander, C.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press (1977)
4. Atencia, M., David, J., Euzenat, J., Napoli, A., Vizzini, J.: Link key candidate extraction with relational concept analysis. Discret. Appl. Math. **273**, 2–20 (2020)
5. Azmeh, Z., Driss, M., Hamoui, F., Huchard, M., Moha, N., Tibermacine, C.: Selection of composable web services driven by user requirements. In: IEEE Int. Conf. on Web Services (ICWS). pp. 395–402. IEEE Computer Society (2011)
6. Azmeh, Z., Huchard, M., Napoli, A., Hacene, M.R., Valtchev, P.: Querying relational concept lattices. In: 8th Int. Conf. on Concept Lattices and Their Applications (CLA). CEUR Workshop Proc., vol. 959, pp. 377–392 (2011)
7. Carbonnel, J., Huchard, M., Nebut, C.: Modelling equivalence classes of feature models with concept lattices to assist their extraction from product descriptions. J. Syst. Softw. **152**, 1–23 (2019)
8. Codocedo, V., Napoli, A.: Formal concept analysis and information retrieval - A survey. In: 13th Int. Conf. Formal Concept Analysis (ICFCA). LNCS, vol. 9113, pp. 61–77. Springer (2015)
9. Dolques, X., Huchard, M., Nebut, C., Reitz, P.: Learning transformation rules from transformation examples: An approach based on relational concept analysis. In: Work. Proc. of the 14th IEEE Int. Enterprise Distributed Object Computing Conf. (EDOCW). pp. 27–32 (2010)
10. Dolques, X., Huchard, M., Nebut, C., Reitz, P.: Fixing generalization defects in UML use case diagrams. Fundam. Informaticae **115**(4), 327–356 (2012)
11. Dolques, X., Le Ber, F., Huchard, M., Grac, C.: Performance-friendly rule extraction in large water data-sets with AOC posets and relational concept analysis. Int. Journal of General Systems **45**(2), 187–210 (2016)
12. Ferré, S., Cellier, P.: Graph-fca: An extension of formal concept analysis to knowledge graphs. Discrete Applied Mathematics **273**, 81–102 (2020)
13. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley Longman Publishing Company (1995)
14. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: 9th Int. Conf. on Conceptual Structures (ICCS). LNCS, vol. 2120, pp. 129–142. Springer (2001)

15. Ganter, B., Wille, R.: Formal Concept Analysis - Mathematical Foundations. Springer (1999)
16. Guédi, A.O., Miralles, A., Huchard, M., Nebut, C.: A practical application of relational concept analysis to class model factorization: Lessons learned from a thematic information system. In: 10th International Conference on Concept Lattices and Their Applications (CLA). CEUR Workshop Proceedings, vol. 1062, pp. 9–20 (2013)
17. Hacene, M.R., 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)
18. Hlad, N., Lemoine, B., Huchard, M., Seriai, A.: Leveraging relational concept analysis for automated feature location in software product lines. In: The ACM SIGPLAN International Conference on Generative Programming: Concepts & Experiences (GPCE), Chicago, IL, USA. pp. 170–183. ACM (2021)
19. Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Relational concept discovery in structured datasets. Ann. Math. Artif. Intell. **49**(1-4), 39–76 (2007)
20. Kasri, S., Benchikha, F.: Refactoring ontologies using design patterns and relational concepts analysis to integrate views: the case of tourism. Int. J. Metadata Semant. Ontologies **11**(4), 243–263 (2016)
21. 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: 15th Int. Conf. on Concept Lattices and Their Applications (CLA). CEUR Workshop Proceedings, vol. 2668, pp. 197–208 (2020)
22. Kötters, J., Eklund, P.W.: Conjunctive query pattern structures: A relational database model for formal concept analysis. Discrete Applied Mathematics **273**, 144–171 (2020)
23. Kouhoué, A.W., Bonavero, Y., Bouétou, T.B., Huchard, M.: Exploring variability of visual accessibility options in operating systems. Future Internet **13**(9), 230 (2021)
24. Mahrach, L., Gutierrez, A., Huchard, M., Keip, P., Marnotte, P., Silvie, P., Martin, P.: Combining implications and conceptual analysis to learn from a pesticidal plant knowledge base. In: 26th Int. Conf. on Conceptual Structures (ICCS). LNCS, vol. 12879, pp. 57–72. Springer (2021)
25. Mimouni, N., Fernández, M., Nazarenko, A., Bourcier, D., Salotti, S.: A relational approach for information retrieval on XML legal sources. In: Int. Conf. on Artificial Intelligence and Law (ICAIL). pp. 212–216. ACM (2013)
26. Moha, N., Hacene, A.R., Valtchev, P., Guéhéneuc, Y.: Refactorings of design defects using relational concept analysis. In: 6th Int. Conf. on Formal Concept Analysis (ICFCA). LNCS, vol. 4933, pp. 289–304. Springer (2008)
27. Nica, C., Braud, A., Le Ber, F.: Exploring Heterogeneous Sequential Data on River Networks with Relational Concept Analysis. In: 23rd Int. Conf. on Conceptual Structures (ICCS). LNAI, vol. 10872, pp. 152–166. Springer (2018)
28. 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)
29. Ouzerdine, A., Braud, A., Dolques, X., Huchard, M., Le Ber, F.: Adjusting the exploration flow in relational concept analysis - an experience on a watercourse quality dataset. In: Advances in Knowledge Discovery and Management, Vol. 9 [Best of EGC 2019]. Studies in Computational Intelligence, vol. 1004, pp. 175–198. Springer (2019)

30. Pérez-Gámez, F., Cordero, P., Enciso, M., López-Rodríguez, D., Mora, Á.: Computing the mixed concept lattice. In: Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU). pp. 87–99. Springer (2022)
31. Poelmans, J., Ignatov, D.I., Kuznetsov, S.O., Dedene, G.: Formal concept analysis in knowledge processing: A survey on applications. Expert Systems with Applications **40**(16), 6538–6560 (2013)
32. Rouane Hacene, A.M., Napoli, A., Valtchev, P., Toussaint, Y., Bendaoud, R.: Ontology Learning from Text using Relational Concept Analysis. In: Int. MCETECH Conf. on e-Technologies (2008)
33. Wajnberg, M.: Analyse relationnelle de concepts : une méthode polyvalente pour l'extraction de connaissance. (Relational concept analysis: a polyvalent tool for knowledge extraction). Ph.D. thesis, Univ. du Québec à Montréal (2020)
34. Wajnberg, M., Valtchev, P., Lezoche, M., Massé, A.B., Panetto, H.: Concept analysis-based association mining from linked data: A case in industrial decision making. In: Joint Ontology Works. 2019 Episode V: The Styrian Autumn of Ontology. CEUR Workshop Proc., vol. 2518. CEUR-WS.org (2019)
35. Wajnberg, M., Lezoche, M., Blondin-Massé, A., Valchev, P., Panetto, H., Tyvaert, L.: Semantic interoperability of large systems through a formal method: Relational concept analysis. IFAC-PapersOnLine **51**(11), 1397–1402 (2018), 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM