RCA as a data transforming method: a comparison with propositionalisation

Xavier Dolques, Kartick Chandra Mondal, Agnès Braud, Marianne Huchard, Florence Le Ber

RCA as a data transforming method: a comparison with propositionalisation

Xavier Dolques[1], Kartick Chandra Mondal[2], Agnès Braud[2], Marianne Huchard[3], and Florence Le Ber[1]

(1) ICube, University of Strasbourg/ENGEES, CNRS
{xavier.dolques, florence.leber}@engees.unistra.fr
(2) ICube, University of Strasbourg, CNRS
{mondal, agnes.braud}@unistra.fr
(3) LIRMM, University of Montpellier 2, CNRS
huchard@lirmm.fr

Abstract. This paper aims at comparing transformation-based approaches built to deal with relational data, and in particular two approaches which have emerged in two different communities: Relational Concept Analysis (RCA), based on an iterative use of the classical Formal Concept Analysis (FCA) approach, and Propositionalisation coming from the Inductive Logic Programming community. Both approaches work by transforming a complex problem into a simpler one, namely transforming a database consisting of several tables into a single table. For this purpose, a main table is chosen and new attributes capturing the information from the other tables are built and added to this table. We show the similarities between those transformations for what concerns the principles underlying them, the semantics of the built attributes and the result of a classification performed by FCA on the enriched table. This is illustrated on a simple dataset and we also present a synthetic comparison based on a larger dataset from the hydrological domain.

1 Introduction

In several applications, data present various characteristics (e.g. many-valued, temporal, spatial) which are not easy to take into account. Relational data in particular are generally transformed into a single table to be processed by data mining methods. In the field of Inductive Logic Programming, propositionalisation approaches (PA) aim at performing such transformations [1]. These approaches can be divided into database-oriented and logic-oriented such as the HiFi method [2]. HiFi allows to build features that are first-order logic conjunctions from related tables. In the field of Formal Context Analysis (FCA, [3]), relational information is addressed by Relational Concept Analysis (RCA, [4]). It has been designed to handle several formal contexts, corresponding to several categories of objects, and several relations between these objects, based on an iterative use of the classical Formal Concept Analysis algorithm. RCA classifies the objects of the different categories in lattices that are connected via relational attributes. The analysis often focuses on a main category of objects, classified in a lattice which is the central point for analyzing data, while navigating towards the other, secondary lattices. Both methods enable us to turn the objects linked to a given object into special attributes, that are propositional features for PA or relational attributes for RCA.

In this paper we propose to compare the two methods, focusing on the semantics of the built attributes, in the context of acyclic data. FCA was used as a common classification method: it was applied on the propositional features obtained by the HiFi method from a given relational dataset and the resulting lattice was compared to the one obtained by the RCA method on the same relational dataset. We detail our comparison on a simple example about pizzas and their ingredients. Another comparison is also performed on a larger dataset from the hydrological domain. The lattices obtained appeared to be isomorphic and allowed to reveal the links between the propositional features in HiFi and the concept generators in RCA.

The paper is organized as follows. Section 2 describes a simple example that is used in Section 3 and 4 to introduce the principles of the RCA and Propositionalisation approaches. Section 5 details the results of the comparison performed both on the simple example and on the real dataset. Related work is described in Section 6. Section 7 concludes and draws some perspectives of this work.

2 A motivating example

The considered objects of our dataset (see Table 1) are people, pizzas, and ingredients. People are farmers described by their current production methodology (organic versus conventional). Pizzas are described by some typology of their shape (thin, thick, calzone). Ingredients are described by their category (fruit/vegetable, meat, fish, dairy). Two relations link these objects: People *prefer* some pizzas, pizzas *have* some ingredients.

A group of people (Juliet, Nancy and Alice) likes at least one pizza containing one dairy ingredient. A subgroup of this group (Nancy and Alice) corresponds to the conventional farmers and we deduce that in this dataset all conventional farmers like at least one pizza containing one dairy ingredient.

For extracting this kind of knowledge from the various relations, it is worth noting that several of them have to be crossed (here the relations `Prefers` and `HasIngredient`). Besides, the group Juliet, Nancy and Alice has initially no pizza in common and no common production methodology, because Juliet is an organic farmer, while Nancy and Alice are conventional farmers. Thus there is no direct reason for grouping these three people. The group Juliet, Nancy and Alice can be formed after two classification steps: (1) the recognition of pizzas Arctic, Lorraine, ThreeCheeses, and FourCheeses as belonging to the group D of pizzas with at least one dairy ingredient; (2) the fact that Juliet, Nancy and Alice like at least one pizza from the D group.

Such a kind of classification is the objective of the two approaches that we study in the following of this paper. This simple dataset can thus be used to exemplify the properties of these two approaches.

Table 1. The dataset

People

Name	ProdMethod
Arthur	OrganicFarmer
John	OrganicFarmer
Alice	ConventionalFarmer
Juliet	OrganicFarmer
Nancy	ConventionalFarmer

Ingredient

IngName	Category
TomatoSauce	FruitVegetable
Cream	Dairy
Onion	FruitVegetable
Bacon	Meat
Salmon	Fish
SoyCream	FruitVegetable
Mozza	Dairy
GoatCheese	Dairy
Emmental	Dairy
FourmeAmbert	Dairy
EggPlant	FruitVegetable
Mushroom	FruitVegetable

Pizza

PizzaName	Shape
Forest	Thick
Occitane	Calzone
ThreeCheeses	Thin
FourCheeses	Thin
Lorraine	Thin
Arctic	Thick

Prefers

Name	PizzaName
Arthur	Forest
John	Occitane
Alice	FourCheeses Lorraine
Juliet	ThreeCheeses Arctic
Nancy	Arctic

HasIngredient

PizzaName	IngName
Forest	SoyCream Mushroom
Occitane	TomatoSauce Onion EggPlant
ThreeCheeses	TomatoSauce Mozza GoatCheese Emmental
FourCheeses	TomatoSauce Cream Mozza GoatCheese Emmental FourmeAmbert
Lorraine	Cream Onion Bacon Mozza
Arctic	TomatoSauce Cream Salmon Mozza

3 Relational Concept Analysis

In this part, the principles of relational concept analysis are presented based on the example described in Section 2. For more details about RCA, the reader is invited to read [5] which refines notations of [4].

The pizza dataset cannot be directly handled by RCA, it must first be transformed. Here, we choose to make a nominal scaling of the three tables People, Pizza and Ingredient to obtain three object-attribute contexts, respectively $\mathcal{K}_{People}$, $\mathcal{K}_{Pizza}$ and $\mathcal{K}_{Ingredient}$. For example, in $\mathcal{K}_{People}$ object-attribute context, objects (G_{People}) are people and attributes (M_{People}) are `OrganicFarmer` and `ConventionalFarmer`. I_{People} contains a pair (p, m) if and only if p has the `ProdMethod` m in People table of the initial dataset, e.g., the pair ($Arthur$, $OrganicFarmer$) belongs to I_{People}. Tables Prefers and HasIngredient give rise to $r_{Prefers}$ and $r_{HasIngredient}$ object-object relations also using a nominal scaling, e.g. $r_{Prefers}$ contains $(Arthur, Forest)$. Finally we obtain a set of contexts and a set of relations between these contexts: $\{\mathcal{K}_{People}, \mathcal{K}_{Pizza}, \mathcal{K}_{Ingredient}\}$, $\{r_{Prefers}, r_{HasIngredient}\}$. More generally, such a structure is called a Relational Context Family and defined as below.

Definition 1 (Relational Context Family (RCF)). *A Relational Context Family (denoted RCF) is a* $(\mathbf{K}, \mathbf{R})$ *pair where:*

– $\mathbf{K} = \{\mathcal{K}_i\}_{i=1,\ldots,n}$ *is a set of* $\mathcal{K}_i = (G_i, M_i, I_i)$ *formal contexts (object-attribute relations), where* G_i *is the set of objects,* M_i *is the set of attributes and* $I_i \subseteq G_i \times M_i$.
– $\mathbf{R} = \{r_j\}_{j=1,\ldots,m}$ *is a set of* r_j *object-object relations where* $r_j \subseteq G_{i_1} \times G_{i_2}$ *for some* $i_1, i_2 \in \{1, \ldots, n\}$.

The principle of RCA consists in integrating object-object relations as new attributes (called *relational attributes*) in formal contexts. A naive approach would be to directly integrate relations as attributes of the form $(relation, targetobject)$, e.g. $(HasIngredient, Mushroom)$, an attribute that could be assigned to the $Forest$ pizza. Such an approach would be able to discover the concept of pizzas with dairies. But it is limited to this one-step deduction and it cannot go beyond. The objective of RCA is to infer classifications based on the composition of several relations, e.g. RCA will be able to group people preferring pizzas having at least one dairy product among their ingredients. This is implemented in RCA via the transformation of the object-object relations into relations between objects of one category, and concepts formed on objects of another category. Such a transformation is made thanks to relational attributes and *scaling operators.* These relational attributes will have the form `q r(C)` where `q` is a *quantifier*, `r` is the relation and `C` is a concept. Theoretically, quantifiers can be chosen within the set $\mathbf{Q} = \{\forall, \exists, \forall\exists, \geq, \geq_q, \leq, \leq_q\}$. The most used quantifiers are:

– the *existential* quantifier ($\exists$) which encodes the fact that an object o is in relation by $\exists r$ with a concept C if $r(o)$ has a non-empty intersection with $Extent(C)$;
– the *strict universal* quantifier ($\forall\exists$) which encodes the fact that an object o is in relation by $\forall\exists r$ with a concept C if $r(o)$ is non-empty and included in the extent of C.

Let us now consider the concept lattices given in Fig. 1, built using any standard algorithm for FCA from the three formal contexts $\mathcal{K}_{People}$, $\mathcal{K}_{Pizza}$, and $\mathcal{K}_{Ingredient}$. In the following we examine the transformation of the pizza-ingredient relation $r_{HasIngredient}$ during its integration as new attributes for describing pizzas. In the lattice of ingredients, $Concept\_Ingredient\_5$ represents the group of dairies. Besides, we observe that all pizzas, except $Forest$ and $Occitane$ pizzas, contain at least one ingredient which is a dairy. This is introduced as a relational attribute $\exists HasIngredient(Concept\_Ingredient\_5)$ shared by $Lorraine$, $Arctic$, $ThreeCheeses$ and $FourCheeses$ pizzas. Now, if we consider people, $Juliet$, $Nancy$ and $Alice$ prefer at least one pizza of this group, and they can be grouped into the concept of people that prefer at least one pizza that contains a dairy ingredient. Furthermore, to illustrate the universal scaling operator, let us have a look at $Concept\_Ingredient\_1$, grouping the fruits and vegetables. $Forest$ and $Occitane$ pizzas have all their ingredients in the extent of this concept. This is introduced as a new relational attribute $\forall\exists HasIngredient(Concept\_Ingredient\_1)$ which can be assigned to $Forest$ and $Occitane$ pizzas (highlighting the concept of vege pizzas). The pizzas that

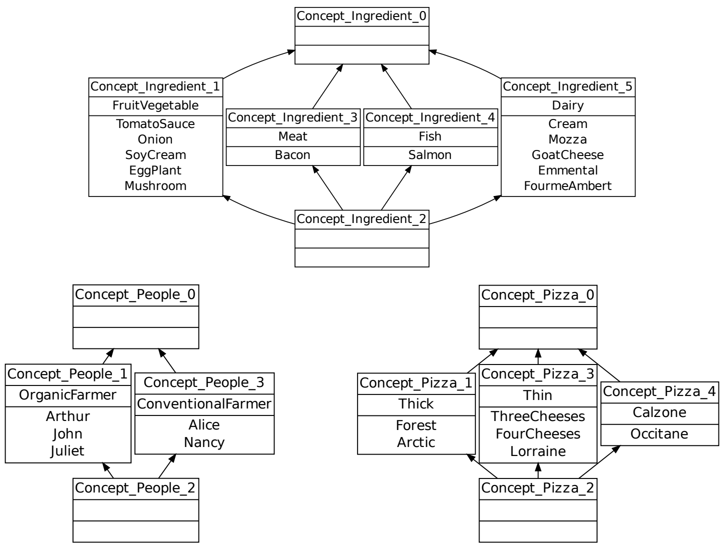

Fig. 1. Lattices for object-attribute relations Ingredient ($\mathcal{L}^0_{Ingredient}$), People ($\mathcal{L}^0_{People}$) and Pizza ($\mathcal{L}^0_{Pizza}$) (step 0 of RCA)

are preferred by Arthur and John are all in the group of vege pizzas, an indication to group these two people.

For defining the scaling operators, a generic function κ is introduced and instantiated with (1) the existential and (2) the strict universal quantifiers:

$$\begin{array}{lllll} \kappa: & \mathbf{Q} & \times\mathbf{R} \times & \bigcup_{i=1,\dots,n} 2^{G_i} & \rightarrow \bigcup_{i=1,\dots,n} 2^{G_i} \\ (1) & \exists & r & Extent(C) & \rightarrow \{o|r(o) \cap Extent(C) \neq \emptyset\} \\ (2) & \forall\exists & r & Extent(C) & \rightarrow \{o|r(o) \subseteq Extent(C) \,\text{and}\, r(o) \neq \emptyset\} \end{array}$$

A scaling operator can now be defined as follows.

Definition 2 (Scaling operator). *Let $\mathcal{K} = (G, M, I)$ be a context, and r a relation, where G is the domain of r; let G_{i_r} be the range of r, $\mathcal{K}_{i_r} = (G_{i_r}, M_{i_r}, I_{i_r})$ another context, and $\mathcal{L}_{i_r}$ a lattice built on $\mathcal{K}_{i_r}$; q denotes a scaling quantifier. The scaling operator $\mathbb{S}_{(r,q),\mathcal{L}_{i_r}}$ over $\mathcal{K}$ yields the derived context $(G^+, M^+, I^+) = \mathbb{S}_{(r,q),\mathcal{L}_{i_r}}(\mathcal{K})$, where:*

- $G^+ = G$,
- $M^+ = \{'q\ r(c)' \mid c \in \mathcal{L}_{i_r}\}$,
- $I^+ = \bigcup_{c \in \mathcal{L}i_r} \kappa(q, r, Extent(c)) \times \{'q\ r(c)'\}$.

The $r_{HasIngredient}$ transformed by the existential scaling, considering the lattice previously built for ingredients (see Fig. 1), is $\mathbb{S}_{(r_{HasIngredient},\exists),\mathcal{L}^0_{Ingredients}}(\mathcal{K}_{Pizza})$. It is shown in Table 2 after the vertical triple bar. The original context $\mathcal{K}_{Pizza}$ can thus be extended with relational attributes representing the relation $r_{HasIngredient}$ between pizzas and ingredients.

Table 2. $\mathcal{K}_{Pizza}$ apposed to existential scaling of $r_{HasIngredient}$. CI stands for 'Concept_Ingredient'

	Thick	**Thin**	**Calzone**	**∃ HasIngredient(CI_0)**	**∃ HasIngredient(CI_1)**	**∃ HasIngredient(CI_2)**	**∃ HasIngredient(CI_3)**	**∃ HasIngredient(CI_4)**	**∃ HasIngredient(CI_5)**
Forest	×			×	×				
Occitane			×	×	×				
ThreeCheeses		×		×	×				×
FourCheeses		×		×	×				×
Lorraine		×		×	×		×		×
Arctic	×			×	×			×	×

Then, for each $\mathcal{K}$ context of $\mathbf{K}$, the *apposition* of $\mathcal{K}$ (denoted by symbol '|') with the respective results of the scaling upon each r_j of $\mathbf{R}$ with G as domain $(1 \leq j \leq k)$, is used to build a new set of concepts (notations are taken from Def. 2). This apposition is the relational extension of the $\mathcal{K}$ context considering a scaling operator mapping ρ and a set of lattices $\mathbf{L}$ which is a union of concept lattices including $\mathcal{L}_{i_{r_j}}$, $1 \leq j \leq k$:

$$\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}) = \mathcal{K} \mid \mathbb{S}_{(r_1,\rho(r_1)),\mathcal{L}_{i_{r_1}}}(\mathcal{K}) \mid \ldots \mid \mathbb{S}_{(r_k,\rho(r_k)),\mathcal{L}_{i_{r_k}}}(\mathcal{K})$$

Table 2 shows this result for $\mathcal{K}_{Pizza}$, when considering $\rho(r_{HasIngredient}) = \exists$ and the lattices of Fig. 1. If an additional relation connecting pizzas to another kind of objects, for example, *IsAppreciatedBy*, connecting pizzas to people had been present in the dataset, then the relational extension of $\mathcal{K}_{Pizza}$ would include the scaling upon *IsAppreciatedBy* too.

By extension, $\mathbb{E}^*_{\rho,\mathbf{L}}(\mathbf{K})$ denotes the relational extension of $\mathbf{K}$, which is composed of all the relational extensions of all $\mathcal{K}_i$ in $\mathbf{K}$ (and $\mathbf{L}$ is a union of concept lattices associated with all ranges of all relations).

$$\mathbb{E}^*_{\rho,\mathbf{L}}(\mathbf{K}) = \{\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_1), \ldots, \mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_n)\}$$

In our example, if we consider only the existential scaling and the lattices of Fig. 1, the relational extension of $\mathbf{K}$ would be composed of the relational extensions of $\mathcal{K}_{People}$, $\mathcal{K}_{Pizza}$ and $\mathcal{K}_{Ingredient}$. The relational extension of $\mathcal{K}_{Ingredient}$ is simply $\mathcal{K}_{Ingredient}$, because there is no outgoing relation. The relational extension of $\mathcal{K}_{Pizzas}$ has been shown in Table 2. The relational extension of $\mathcal{K}_{People}$ is $\mathcal{K}_{People}$ apposed to $\mathbb{S}_{(r_{Prefers},\exists),\mathcal{L}^0_{Pizza}}(\mathcal{K}_{People})$.

Now a whole construction process consists in building a finite sequence of contexts and concept lattices associated with $(\mathbf{K}, \mathbf{R})$ and ρ. The last sequence is obtained when the fix point is reached. The first set of contexts (step 0) is

$\mathbf{K}^0 = \mathbf{K}$. The contexts of step p are used to build the associated concept lattices. The $\mathbf{L}_p$ set composed of the lattices at step p is used to calculate the relational extension. The set of contexts at step $p+1$ is defined using the relational extension: $\mathbf{K}^{p+1} = \mathbb{E}^*_{\rho,\mathbf{L}_p}(\mathbf{K}_p)$.

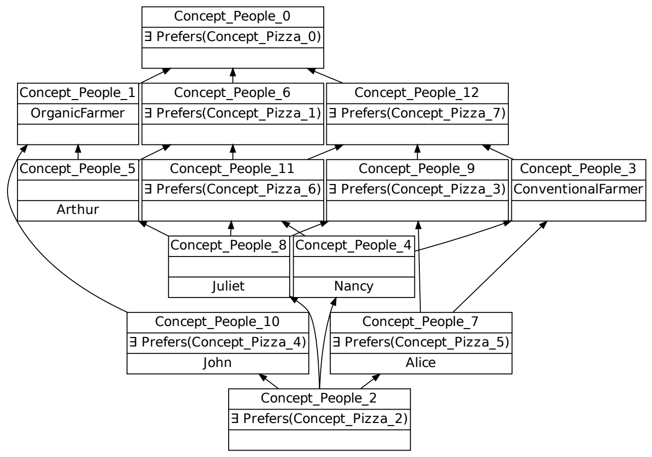

Fig. 2. Lattice of people ($\mathcal{L}^3_{People}$) (step 3 of RCA)

For our example, the fix point is obtained after three steps. The lattice for ingredients is the same during all the process (see Fig. 1). The lattices for people and pizzas are shown in Fig. 2 and 3. In $\mathcal{L}^3_{Pizza}$ lattice, $Concept\_Pizza\_7$ represents the group of pizzas which contain at least one ingredient which is a dairy. In $\mathcal{L}^3_{People}$ lattice, $Concept\_People\_12$ represents the group of people which prefer at least one pizza which contains at least one dairy ingredient. Figure 4 presents the three concepts involved in these groups of people, pizzas and ingredients respectively.

4 Propositionalisation: the HiFi method

Propositionalisation has emerged within the field of Inductive Logic Programming (ILP) [6]. Initially ILP was concerned with learning logic programs, and ILP techniques have then been applied in relational data mining. In ILP, learning is performed directly in the first-order logic setting, so that the space to search is intractable when data are numerous. Propositionalisation [1] was proposed as a mean to reduce this complexity. The idea is to shift from a representation in first-order logic to an attribute-value one. This is usually done in two steps: (1) computation of new attributes, called features, for the attribute-value representation (2) computation of the extensions (the values in the resulting propositional table). For some techniques, the two steps are performed at the same time. It

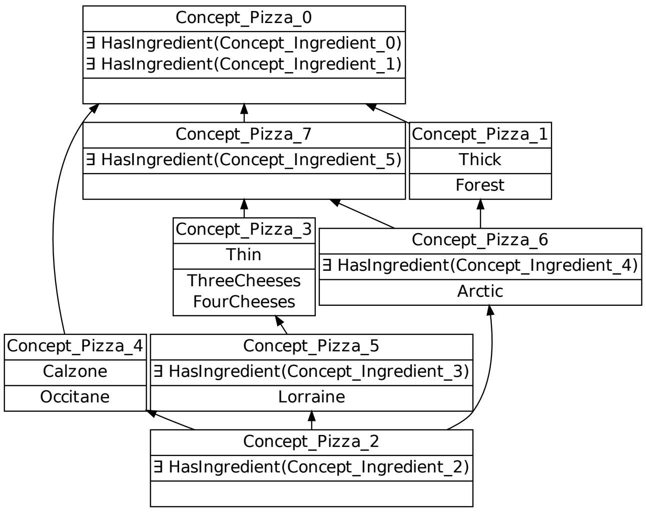
Concept_Pizza_0
∃ HasIngredient(Concept_Ingredient_0)
∃ HasIngredient(Concept_Ingredient_1)
Concept_Pizza_7
∃ HasIngredient(Concept_Ingredient_5)
Concept_Pizza_1
Thick
Forest
Concept_Pizza_3
Thin
ThreeCheeses
FourCheeses
Concept_Pizza_6
∃ HasIngredient(Concept_Ingredient_4)
Arctic
Concept_Pizza_4
Calzone
Occitane
Concept_Pizza_5
∃ HasIngredient(Concept_Ingredient_3)
Lorraine
Concept_Pizza_2
∃ HasIngredient(Concept_Ingredient_2)

Fig. 3. Lattice of pizzas ($\mathcal{L}^3_{Pizza}$) (step 3 of RCA)

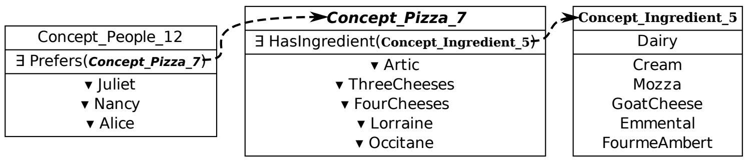

Fig. 4. Chained concepts for people (from $\mathcal{L}^3_{People}$), pizzas (from $\mathcal{L}^3_{Pizza}$) and ingredients (from $\mathcal{L}^3_{Ingredient}$) at step 3 of RCA (Objects in the extent that do not belong to the simplified extent are signaled by ▼)

is then possible to apply one of the many efficient propositional systems on the propositional table. The logic-oriented approach HiFi [2] produces such a propositional table that can be then processed by FCA. Other logic-based approaches exist but we have chosen HiFi for its similarities with RCA.

A database can be seen as a couple $\mathcal{DB} = (\mathcal{R}, \mathcal{C})$, where $\mathcal{R}$ is a set of relations $r_i(a_{i_1}, ..., a_{i_n})$ and $\mathcal{C}$ is a set of reference constraints on some attributes of these relations $(c_i : a_{j_k} \rightarrow a_{l_m})$ (foreign keys). The database representation is directly transformed into first-order logic, each relation becoming a predicate.

In propositionalisation, a main relation, let say r_1, is chosen that corresponds to the description of the object of interest. The other relations are then called secondary. The aim of propositionalisation is to generate features that capture the relevant information from the secondary relations to enrich the description of objects from the relation r_1. For example, if *People* is chosen as main table, *People* is the object of interest, that is the one on which we focus our study. $Pizza$, $Ingredient$, $HasIngredient$ and $Prefers$ are the secondary tables. Features will capture information on objects in those four tables that are linked to

People. The two last relations will allow to work on relations between the different objects represented in the database. $\mathcal{C}$ gives the links between the relations.

HiFi produces features which are function-free first-order conjunctions. Those features are based on a template given by the user and belong to a specific class of features called hierarchical features. A template defines the literals that may appear in a feature, as well as some constraints on the arguments of a literal. Let T be a template on the pizzas example:

$T = People(-Name), Prefers(+Name, -PizzaName), Pizza(+PizzaName, \#Shape), Pizza(+PizzaName, !Shape), HasIngredient(+PizzaName, -IngName), Ingredient(+IngName, \#Category)$.

In this template, $Name$, $PizzaName$, $Shape$, $IngName$ and $Category$ act as types and indicate which arguments may share a variable. We can also notice modes: + (intput), - (output), # (constant) and ! (ignored). The input mode means that the argument will be a variable and it will be instantiated. The output mode indicates an argument which is a variable that receives an already instantiated value. At a position with a # mode, the argument should be a constant. A feature contains literals of the template, moreover any variable that appears as an input/output must appear in the feature as an output/input, except if the variable occurs with a ! mode.

Templates in HiFi are hierarchical. This is obtained by ensuring that: (1) every literal has at most one input argument, (2) there is a partial irreflexive order on types implying that type $t \prec$ type t' whenever t appears as an input and t' as an output in some literal. The above template T is hierarchical: we can check that any literal has at most one input argument and there is no pair of types (t, t') such that there exists a literal where t appears as an input argument and t' as an output argument, and another literal where it is the contrary.

A hierarchical feature is based on a hierarchical template and has exactly one root (a literal with only output variables). It can be represented as a tree where each literal l_i is a node n_i, and an edge between n_i and n_j indicates that a variable has an output occurrence in l_i and an input occurrence in l_j.

HiFi avoids generating redundant features. Indeed, we can define equivalence classes among the set of possible features, which correspond to features having the same extension (they have the same values for all objects). HiFi generates a set of features containing one representative feature for each equivalence class, the one chosen being the smallest in the equivalence class. With template T, HiFi outputs the following set of features on the pizzas dataset (the _ notation comes from the ! mode):

$F_1 : People(A), Prefers(A, B), HasIngredient(B, C), Ingredient(C, Dairy)$
$F_2 : People(A), Prefers(A, B), HasIngredient(B, C), Ingredient(C, Fish)$
$F_3 : People(A), Prefers(A, B), HasIngredient(B, C), Ingredient(C, Meat)$
$F_4 : People(A), Prefers(A, B), Pizza(B, Calzone)$
$F_5 : People(A), Prefers(A, B), Pizza(B, Thick)$
$F_6 : People(A), Prefers(A, B), Pizza(B, Thick), Prefers(A, C), Pizza(C, Thin)$
$F_7 : People(A), Prefers(A, B), Pizza(B, Thin)$
$F_8 : People(A), Prefers(A, B), Pizza(B, \_)$

The corresponding propositional table is shown in Table 3. In this table, *ProdMethod* is a proper attribute of the object of interest *People* and F_i are boolean features generated by HiFi to bring relational information from the secondary tables, and thus enrich the description of People. For example, F_1 is true for people who prefers at least one pizza with ingredients of the dairy category, it is false otherwise.

Table 3. Propositional table

	ProdMethod	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
Arthur	OrganicFarmer	-	-	-	-	+	-	-	+
John	OrganicFarmer	-	-	-	+	-	-	-	+
Alice	ConventionalFarmer	+	-	+	-	-	-	+	+
Juliet	OrganicFarmer	+	+	-	-	+	+	+	+
Nancy	ConventionalFarmer	+	+	-	-	+	-	-	+

5 Methods comparison

5.1 Discussion on the example

On the one hand, the scope of the propositionalisation approach extends to the building of features into a propositional table. On the other hand, the RCA approach goes one step further by building concept lattices from a relational extension. Figure 5 describes both approaches in parallel and highlights the comparison points. The left part of the figure stands for the data transformation part of the processes where relational tables are transformed into single propositional tables. The right part of the figure stands for a propositional algorithm, here FCA. To compare both approaches, we find relevant to consider:

– the *people* relational extension (together with the concept lattices) with the propositional table;
– the *people* concept lattice (together with the other concept lattices) with the concept lattice built from the propositional table.

The propositional table describes a binary relation in the same way as a formal context. Objects are the same in the context and in the propositional table and the attributes are the features found by the propositionalisation algorithm and the initial attributes (here *ProdMethod*). A pair (o, a) is in the incidence relation if a is an initial attribute and o owns that initial attribute or if a is a feature and o is described by it. Thus, it is straightforward to build a concept lattice from a propositional table. The lattice from Fig. 6 has been built from Table 3.

This lattice structure is isomorphic to the one presented in Fig. 2 as they have the same set of concept extents. Thus it appears relevant to study the correspondences between concept intents as done below.

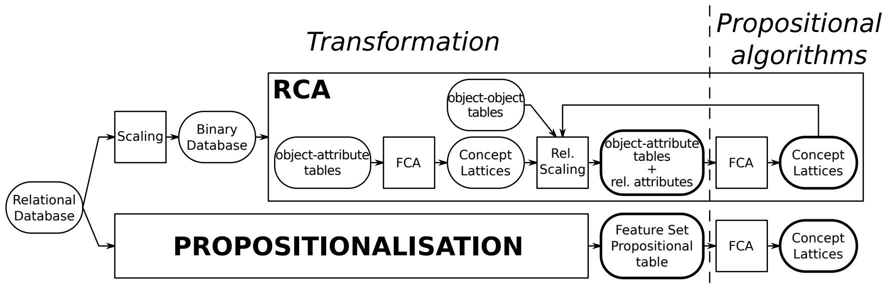

Fig. 5. RCA and propositionalisation processes described in parallel.

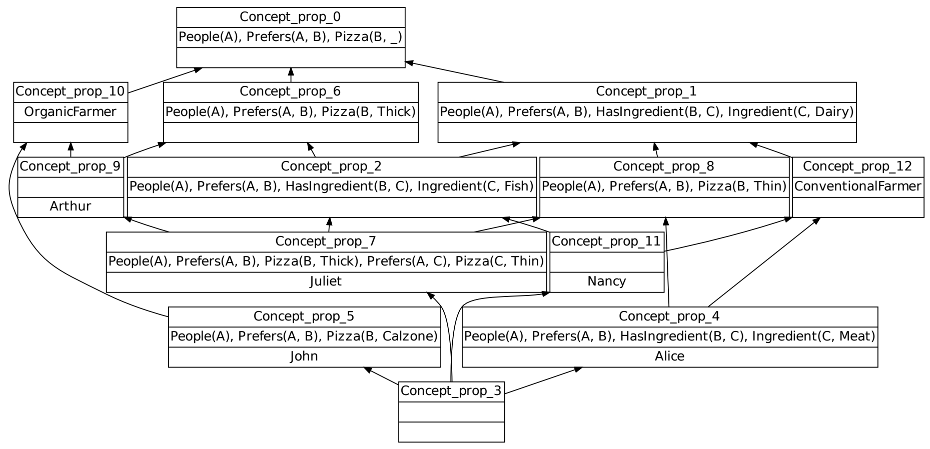
Concept_prop_0
People(A), Prefers(A, B), Pizza(B, _)
Concept_prop_10
OrganicFarmer
Concept_prop_6
People(A), Prefers(A, B), Pizza(B, Thick)
Concept_prop_1
People(A), Prefers(A, B), HasIngredient(B, C), Ingredient(C, Dairy)
Concept_prop_9
Arthur
Concept_prop_2
People(A), Prefers(A, B), HasIngredient(B, C), Ingredient(C, Fish)
Concept_prop_8
People(A), Prefers(A, B), Pizza(B, Thin)
Concept_prop_12
ConventionalFarmer
Concept_prop_7
People(A), Prefers(A, B), Pizza(B, Thick), Prefers(A, C), Pizza(C, Thin)
Juliet
Concept_prop_11
Nancy
Concept_prop_5
People(A), Prefers(A, B), Pizza(B, Calzone)
John
Concept_prop_4
People(A), Prefers(A, B), HasIngredient(B, C), Ingredient(C, Meat)
Alice
Concept_prop_3

Fig. 6. The concept lattice $\mathcal{L}_{prop}$ of people described by features and proper attributes

By considering concept extents, `Concept_prop_0` from lattice $\mathcal{L}_{prop}$ can be mapped to `Concept_People_0` from lattice $\mathcal{L}^3_{people}$. `Concept_prop_0` has for sole feature $People(A), Prefers(A, B), Pizza(B, \_)$ ("people preferring at least one pizza of any shape"). `Concept_People_0` has for sole relational attribute $\exists Prefers($`Concept_Pizza_0`$)$. `Concept_Pizza_0` has 2 relational attributes: $\exists HasIngredient($`Concept_Ingredient_0`$)$ ("pizza having at least one ingredient") and $\exists HasIngredient($`Concept_Ingredient_1`$)$ (pizza having at least one ingredient of the category fruit or vegetable). Hence, `Concept_People_0` is the concept of people preferring at least one pizza with at least one fruit or vegetable (i.e. any pizza in the current dataset). HiFi's goal is to keep the shortest feature describing all the objects and that can be written with the chosen template. It is sufficient to say that "people prefer at least one pizza of any shape" to describe all the people in the dataset and nothing shorter can be written with the current template.

`Concept_prop_8` from lattice $\mathcal{L}_{prop}$ can be mapped to `Concept_People_9`. `Concept_prop_8` groups *Alice* and *Juliet* that own the following features:

1. $People(A), Prefers(A, B), Pizza(B, Thin)$ which is in the proper intent
2. $People(A), Prefers(A, B), HasIngredient(B, C), Ingredient(C, Dairy)$
3. $People(A), Prefers(A, B), Pizza(B, \_)$

`Concept_People_9` also groups *Alice* and *Juliet* and owns the following relational attributes:

1. $\exists Prefers($`Concept_Pizza_3`$)$ where `Concept_Pizza_3` groups "thin pizzas". This attribute is in the proper intent of `Concept_People_9`
2. $\exists Prefers($`Concept_Pizza_7`$)$ where `Concept_Pizza_7` groups "pizzas which contain at least one dairy ingredient"
3. $\exists Prefers($`Concept_Pizza_0`$)$ where `Concept_Pizza_0` groups "all pizzas that contain at least one ingredient and at least one fruit or vegetable ingredient"

The mapping between `Concept_prop_8` and `Concept_people_9` relies on the mapping between the feature which is the proper intent of `Concept_prop_8` and the relational attribute that generates the construction of `Concept_people_9`.

`Concept_prop_7` describes *Juliet* and adds to the features inherited from `Concept_prop_8` the attribute *OrganicFarmer* and the following features:

1. $People(A), Prefers(A, B), Pizza(B, Thick), Prefers(A, C), Pizza(C, Thin)$
2. $People(A), Prefers(A, B), HasIngredient(B, C), Ingredient(C, Fish)$
3. $People(A), Prefers(A, B), Pizza(B, Thick)$

`Concept_people_8` owns the attribute *OrganicFarmer* and the following relational attributes:

1. $\exists Prefers($`Concept_Pizza_3`$)$
2. $\exists Prefers($`Concept_Pizza_7`$)$
3. $\exists Prefers($`Concept_Pizza_0`$)$
4. $\exists Prefers($`Concept_Pizza_6`$)$
5. $\exists Prefers($`Concept_Pizza_1`$)$

The proper intent of `Concept_people_8` is empty. The minimal generators (i.e. the smallest by inclusion subsets of the intent which have the intent as image by the closure function) [7] of `Concept_people_8` are:

– $\{\exists Prefers(Concept\_Pizza\_6), OrganicFarmer\}$
– $\{\exists Prefers(Concept\_Pizza\_7), OrganicFarmer\}$
– $\{\exists Prefers(Concept\_Pizza\_3), OrganicFarmer\}$
– $\{\exists Prefers(Concept\_Pizza\_3), \exists Prefers(Concept\_Pizza\_6)\}$
– $\{\exists Prefers(Concept\_Pizza\_3), \exists Prefers(Concept\_Pizza\_1)\}$

If we discard the first three generators as they contain *OrganicFarmer* which is initially present in the main table for HiFi, we find 2 minimal generators. By replacing the references to other concepts by a generator of these concepts we obtain respectively $\{\exists Prefers(Thin), \exists Prefers(\exists HasIngredient(Fish))\}$ and $\{\exists Prefers(Thin), \exists Prefers(Thick)\}$. In `Concept_prop_7`, the feature of the proper intent is related to the second expression as it is the shortest one. Both `Concept_pizza_3` and `Concept_pizza_1` have a unique generator, respectively the attributes `thick` and `thin`.

The link between a concept c_{RCA} from $\mathcal{L}^3_{people}$ and a concept c_{prop} with same extents appears to reside in the link between a concept generator of c_{RCA} and the feature from the proper intent of c_{prop}. The goal of both approaches can be seen as opposite. While HiFi will tend to provide the shortest description that can discriminate a concept from any other one, RCA will provide the most complete description of a concept.

5.2 Evaluation on a real dataset

We rely on a part of the Fresqueau database, representing data from Alsatian streams and water areas (North-East of France) [8]. The data are either issued from samples (e.g. biological data collected on stream sites), synthetic data (e.g. stream typology, land cover) or general information issued from the literature (e.g. information about the aquatic species living in the streams). More precisely in this paper we work with three many-valued tables. The first one describes 20 stream sites. The second table gives the level of population for 65 macro-invertebrates collected on these 20 sites. The third one describes the macro-invertebrates with 3 different life traits, i.e. their characteristics and functioning (maximal size, aquatic state and reproduction mode), each life trait being represented by several modalities (e.g. for the life trait maximal size there are 7 possible modalities going from less than 0.25cm to more than 8cm) and affinity values. The total number of the modalities for all life traits is 19.

This dataset has been processed by HiFi and RCA (with the $\exists$ scaling quantifier). HiFi template and RCA relational schema define the analysis framework. The following template is used for HiFi: *[Station(-s), presence(#abundance,+s, -macroInv), presence(#abundance, +s, !macroInv), affinity(#level,+macroInv, #modality), affinity(#level,+macroInv,!modality)]*. Accordingly, the relational schema for RCA has 3 formal contexts: *Station*, *MacroInv*, and *Modality* and 6 object-object relations: *abundance-1*, *abundance-2*, and *abundance-3* from *Station* to *MacroInv* and *affinity-1*, *affinity-2*, and *affinity-3* from *MacroInv* to *Modality*.

We found respectively 13460 features and 13461 concepts in the Station lattice. The extent of each feature is the extent of a concept. The additional concept is the bottom concept of the lattice, with an empty extent. So we verified that for each feature can be associated a concept and that the lattice obtained from the propositional table and the Station lattice are isomorphic.

6 Related work

Data transformation is a main issue for all classification or automatic learning methods, when dealing with complex or numerous data. Scaling operators are used in FCA for transforming many-valued contexts into binary ones [3]. Such an approach was also used to analyze complex data about life traits of aquatic plants [9]. Statistical metrics can also be used for helping the transformation, e.g. the χ^2 distance was used for selecting the best scaling operator upon a numerical context [10]. This last idea can be related to the metrics used to design decision trees. A comparison between decision trees and dichotomic lattices (i.e. lattices based on complemented contexts) has been presented in [11]. It was proven that the lattice contained all the trees built on the same context.

In [12], many-valued contexts are transformed into a family of formal contexts (under the guidance of a user objective) which is called the power context family (this notion has been introduced in [13]). It represents all the k-ary relations on the object set. From the concept lattices built on the formal contexts of the power context family, concept graphs are extracted which, in turn, are organized into a lattice. In [14], another approach for obtaining concept graphs is presented, that relies on temporal concept analysis, where the conceptual scales are used instead of the concept lattices of the k-ary relations. In these references, there is no use of different scaling operators and a single-step construction is done (comparatively to the iterative approach of RCA). In [12], graphs connecting objects are classified, while in RCA, objects are classified depending on their relations to other objects.

Relational data have been transformed into logical formulae within the framework of logical concept analysis [15]. Object contexts are combined with relational contexts and equipped with a combined logic. Relational attributes are defined as follows: $(\exists r.f)(x) =_{def} \exists x'.(r(x,x') \wedge f(x))$. The concepts' intents of the resulting lattice contain either classical attributes (f) or relational attributes $(\exists r.f)$. Meta-relations are also built for navigating from a concept to another. Contrarily to RCA, no iteration is performed. In [16], authors propose a method for computing a basis of general concept inclusions in Description Logics $\mathcal{EL}_{gfp}$ where cyclic concept definition has close connections with RCA.

In [17], authors aim at redesigning a database schema. To this end, the database schema is encoded in a formal context and a kind of relational scaling is done in order to represent foreign keys. Here we do not work at the schema level, but at the object level, and the links between objects, rather than the relations between the tables are the focus of the transformation.

Boolean Factor Analysis is applied to multi-relational data in [18]. Their relational factors are tuples of boolean factors extracted independently from the various data tables. In this approach, several schemas of connection can be applied that are similar to the scaling operators of RCA (like existential or universal). Compared to RCA, the boolean factors (that are included in relational factors) are only a part of the formal concepts that could be built from the object-attribute tables, while in RCA all such formal concepts are initially considered. Besides, the process does not iterate.

The authors of [19] address the navigation of SPARQL query answers in concept lattices. They propose a transformation of an RDF graph to a formal context where relations are encoded as attributes. The concept lattice helps analyzing the query answers through their classification.

Reference [20] also considers objects connected by relations. It introduces a Galois connection (and the derived concept lattice) which associates a table (variables and the corresponding tuples) to a description that takes the form of a *windowed s-structure*. Such a windowed s-structure (designed to be a form of a query) is roughly a graph with edges labelled by the relations and with some nodes labelled by variables. There are some similarities between the windowed s-structures, the features and the relational attributes (when they are unfolded). In RCA, concepts correspond to tables with only one variable and finding the equivalent of the tables with more than one variable would rely on navigating on (potentially) several lattices and considering queries like in [21]. Besides, in [20] only existential queries are expressed and there is no iteration, thus no possibility to progressively find the concepts.

7 Conclusion

Several approaches exist in the literature to extract knowledge from relational data, using different data transformation methods. In this paper, we focus on two approaches, namely Relational Concept Analysis and Propositionalisation, which we compare on a small example and on a real dataset. We identify similarities in their objectives and between the features of the Propositionalisation approach and the generators in FCA approach. As future work we would like to evaluate the two approaches on other datasets to confirm the practical feasibility and the similar results, using different tunings including step number (for RCA), frequency or feature literal maximum number (for propositionalisation). We also plan to continue exploring the links between features and generators and in general the theoretical and practical advantages and limits of both approaches. In particular, we will study how other scaling operators used in the RCA framework (universal or involving cardinality restrictions) and cyclic schemas can be considered with the propositionalisation approach point of view. From this research, we expect to define a combined methodology that would improve the efficiency of knowledge extraction in relational data, for example by injecting HiFi results in RCA, or using relational attributes obtained at a given RCA step as information for HiFi.

Acknowledgement. This work was funded by ANR11_MONU14 Fresqueau. We acknowledge Corinne Grac (LIVE) for advices about the hydrological dataset.

References

1. Lachiche, N.: Propositionalization. In Sammut, C., Webb, G.I., eds.: Encyclopedia of Machine Learning. Springer (2010) 812–817
2. Kuželka, O., Železný, F.: HiFi: Tractable Propositionalization through Hierarchical Feature Construction. In: Late Breaking Papers, the 18th Int. Conf. on Inductive Logic Programming. (2008) 1–6
3. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer Verlag (1999)

4. Hacene, M.R., Huchard, M., Napoli, A., Valtchev, P.: Relational concept analysis: mining concept lattices from multi-relational data. Ann. Math. Artif. Intell. **67**(1) (2013) 81–108
5. Hacene, M.R., Huchard, M., Napoli, A., Valtchev, P.: Soundness and completeness of relational concept analysis. In: 11th Int. Conf. on Formal Concept Analysis, ICFCA 2013, Dresden, Germany. LNCS 7880 (2013) 228–243
6. Muggleton, S., Raedt, L.D.: Inductive logic programming: Theory and methods. Journal of Logic Programming **19**(20) (1994) 629–679
7. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD Int. Conference on Management of Data. (1993) 207–216
8. Grac, C., Le Ber, F., Braud, A., Trémolières, M., Bertaux, A., Herrmann, A., Manné, S., Lafont, M.: Programme de recherche-développement *Indices* – rapport scienfique final. Contrat pluriannuel 1463 de l'Agence de l'Eau Rhin-Meuse, LHYGES – LSIIT – ONEMA – CEMAGREF (2011)
9. Bertaux, A., Le Ber, F., Braud, A., Trémolières, M.: Identifying ecological traits: a concrete FCA-based approach. In: 7th Int. Conf. on Formal Concept Analysis, ICFCA 2009, Darmstadt, Germany. LNAI 5548 (2009) 224–236
10. Hereth, J., Stumme, G., Wille, R., Wille, U.: Conceptual knowledge discovery and data analysis. In: 8th Int. Conf. on Conceptual Structures, ICCS'00, Darmstadt, Germany. LNAI 1867 (2000) 421–437
11. Guillas, S., Bertet, K., Ogier, J.M., Girard, N.: Some links between decision tree and dichotomic lattice. In: 8th Int. Conf. on Concept lattices and applications, CLA 2008, Olomouc, Czech Republic. (2008) 193–205
12. Prediger, S., Wille, R.: The Lattice of Concept Graphs of a Relationally Scaled Context. In: 7th Int. Conf. on Conceptual Structures, ICCS'99, Blacksburg, Virginia. LNCS 1640, Springer (1999) 401–414
13. Wille, R.: Conceptual Graphs and Formal Concept Analysis. In: 5th Int. Conf. on Conceptual Structures, ICCS'97. LNCS 1257 (1997) 290–303
14. Wolff, K.E.: Relational Scaling in Relational Semantic Systems. In: 17th Int. Conf. on Conceptual Structures, ICCS 2009. LNCS 5662 (2009) 307–320
15. Ferré, S., Ridoux, O., Sigonneau, B.: Arbitrary Relations in Formal Concept Analysis and Logical Information Systems. In: 13th Int. Conf. on Conceptual Structures, ICCS'05, Kassel, Germany. LNAI 3596, Springer (2005) 166–180
16. Baader, F., Distel, F.: A finite basis for the set of $\mathcal{EL}$-implications holding in a finite model. In: Formal Concept Analysis, 6th Int. Conf., ICFCA, LNCS 4933. (2008) 46–61
17. Stanley, R., Astudillo, H., Codocedo, V., Napoli, A.: A Conceptual-KDD Approach and its Application to Cultural Heritage. In: 10th Int. Conf. on Concept Lattices and Their Applications, CLA. CEUR Workshop Proceedings 1062 (2013) 163–174
18. Krmelova, M., Trnecka, M.: Boolean Factor Analysis of Multi-Relational Data. In: 10th Int. Conf. on Concept Lattices and Their Applications, CLA 2013, La Rochelle, France. CEUR Workshop Proceedings 1062 (2013) 187–198
19. Chekol, M.W., Napoli, A.: An FCA Framework for Knowledge Discovery in SPARQL Query Answers. In: Int. Semantic Web Conference (Posters & Demos), ISWC 2013, Sydney, Australia. CEUR Workshop Proceedings 1035 (2013) 197–200
20. Kötters, J.: Concept Lattices of a Relational Structure. In: 20th Int. Conf. on Conceptual Structures, ICCS 2013, Mumbai, India. LNCS 7735 (2013) 301–310
21. 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, Nancy, France. CEUR Workshop Proceedings 959 (2011) 377–392