Relational Concept Analysis: a synthesis and open questions

Marianne Huchard

Relational Concept Analysis: a synthesis and open questions

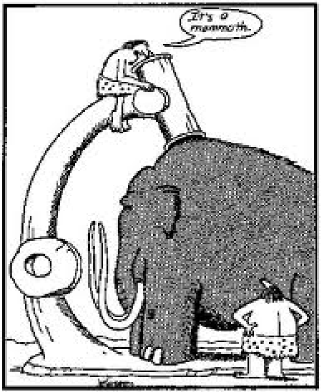
Marianne Huchard

LIRMM, CNRS & Université Montpellier 2, France

FCA4AI - 28 august 2012

Outline

Relational Concept Analysis (RCA)



Early microscope

http://www.mrnorton.com/Chemistry/Cartoons/FarSideMicroscope.gif

Relational Concept Analysis (RCA)

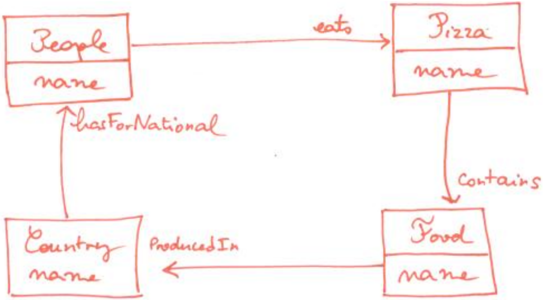
- Extends the purpose of FCA for taking into account links between objects
- Main principles:
  - a relational model based on the entity-relationship model
  - integrate relations between objects as relational attributes
  - iterative process
- RCA provides a *concept lattice family*
- Produced structures can be represented as ontology concepts within a knowledge representation formalism such as description logics (DLs).

Work with A. Napoli, M.A. Rouane-Hacène, C. Roume, P. Valtchev

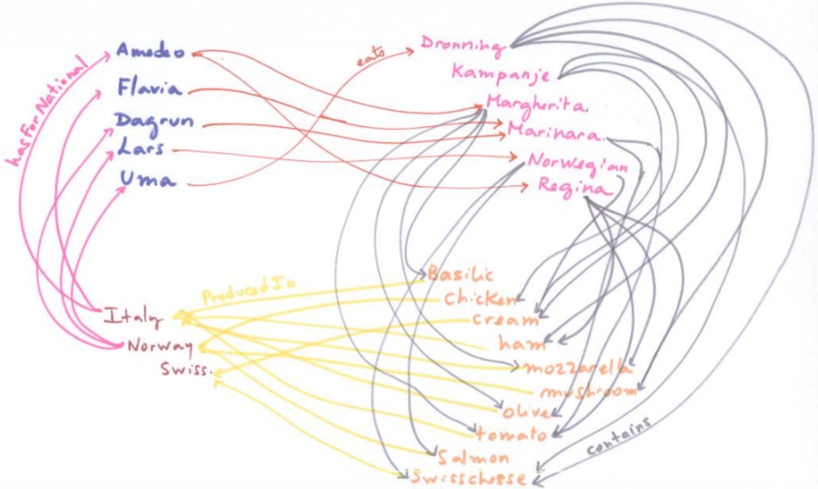
Relational Concept Analysis (RCA)

A relational model based on the entity-relationship model

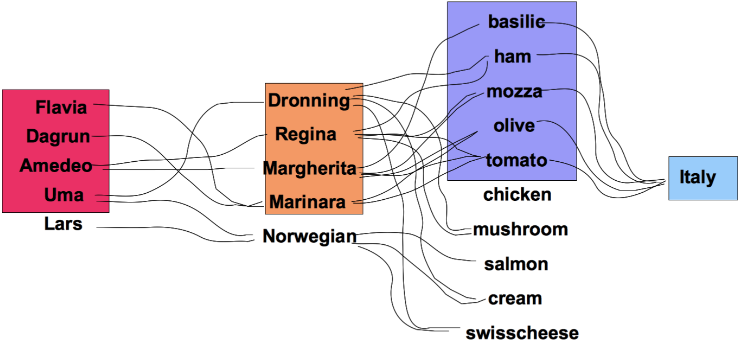
Pizza story



Objects and links



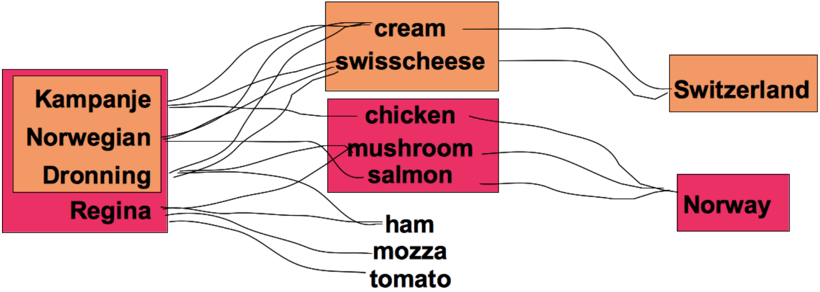
Relevant groups of objects



People who eat at least one pizza containing at least one ingredient produced in Italy

The group is formed using relation compositions and objects far from initial objects

Relevant implications



In Pizzas:
contains at least an ingredient produced in Switzerland
$\Rightarrow$ contains at least an ingredient produced in Norway

Relational Context Family (RCF)

A RCF $\mathcal{F}$ is a pair (K, R) with:

- K is a set of object-attribute contexts $K_i = (O_i, A_i, I_i)$
- R is a set of object-object contexts $R_j = (O_k, O_l, I_j)$,
  - (O_k, O_l) are the object sets of formal contexts $(K_k, K_l) \in K^2$
  - $I_j \subseteq O_k \times O_l$
  - K_k is the *source/domain context*, K_l is the *target/range context*.
  - we may have $K_k = K_l$.

Pizza RCF

$K = K_{People}, K_{Pizza}, K_{Ingredient}, K_{Country}$

$R = R_{eats}, R_{contains}, R_{producedIn}, R_{hasCitizen}$

Example of object-attribute context $K_{Ingredient}$

$K_{Ingredient} = (O_{Ingredient}, A_{Ingredient}, I_{Ingredient})$ Here object (rows) are described by identifiers (columns), more relevant attributes can be used

$I_{Ingredient}$	basilic	chicken	cream	ham	mozzarella	mushroom	olive	tomato	salmon	swisscheese
basilic	×									
chicken		×								
cream			×							
ham				×						
mozzarella					×					
mushroom						×				
olive							×			
tomato								×		
salmon									×	
swisscheese										×

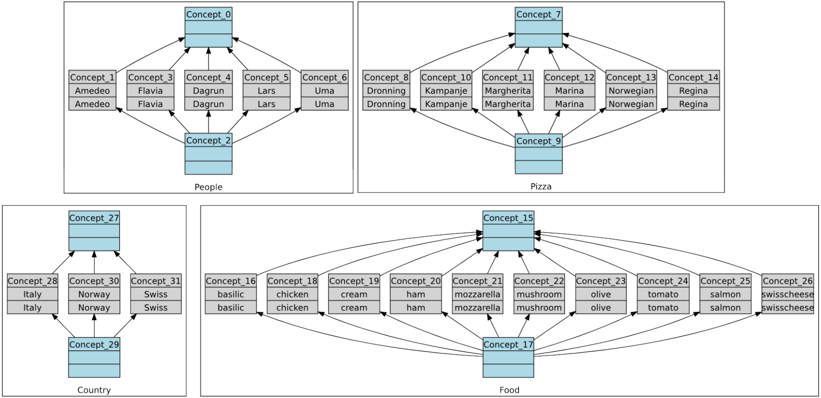
Example of object-object context $R_{contains}$

$$R_{contains} = (O_{Pizza}, O_{Ingredient}, I_{contains})$$

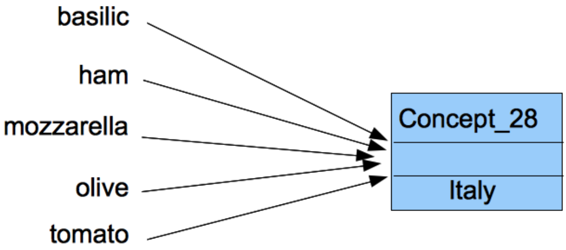
$I_{contains}$	basilic	chicken	cream	ham	mozzarella	mushroom	olive	tomato	salmon	swisscheese
Dronning			×	×		×				×
Kampanje		×	×							×
Margherita	×				×		×	×		
Marina							×	×		
Norwegian			×						×	×
Regina				×	×	×		×		

RCA - Initial Lattice building

At the beginning, only the object-attribute contexts are used to build the foundation of the concept lattice family



RCA - Introducing relations as relational attributes

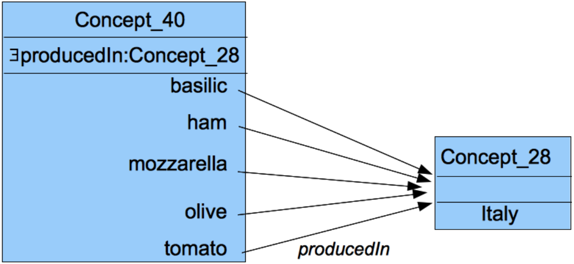


Because basilic, ham, mozzarella, olive and tomato are connected via *producedIn* to Concept_28 extent

basilic, ham, mozzarella, olive, tomato **own** the relational attribute $\exists$*producedIn.Concept_28*

RCA - Introducing relations as relational attributes

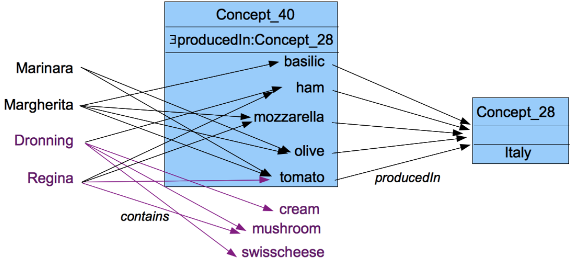
At a further step ...



basilic, ham, mozzarella, olive, tomato grouped because they **own** the relational attribute ∃*producedIn*.*Concept_28*

RCA - Introducing relations as relational attributes

At a further step ...

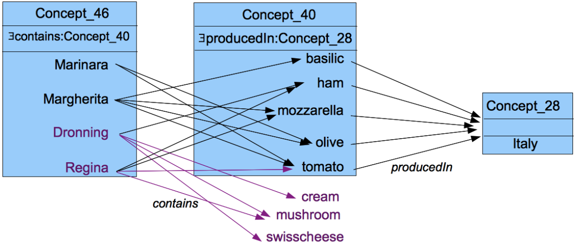


Dronning, Regina, Margherita and Marinara are connected via *contains* to Concept_40 extent

several connection strengths: All are connected to at least one italian ingredient; Margherita and Marinara are besides connected only to italian ingredients; Margherita and Regina are connected to more than 3 italian ingredients, etc.

RCA - Introducing relations as relational attributes

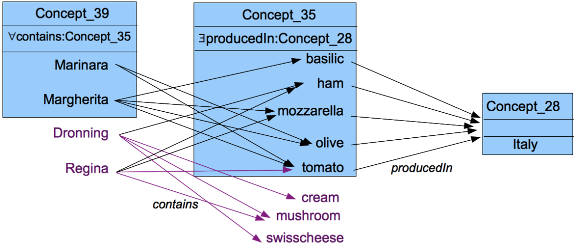
At a further step ...



Dronning, Regina, Margherita and Marinara are grouped, they own $\exists contains.Concept\_40$

RCA - Introducing relations as relational attributes

Alternatively ...



Margherita and Marinara are grouped, they own
$\forall contains.Concept\_35$
(in the whole process the concept numbers change, this is explained later)

RCA - Introducing relations as relational attributes

Relational scaling is the process by which links are established between objects and concepts.

For each relational context $R_j = (O_k, O_l, I_j)$, a scaled context $R_j^* = (O_k, A, I_j)$ is created.

- A is a set of relational attributes $a = S\ R_j.C$, where C is in the concept set of a lattice built on objects of O_l, denoted by $\mathcal{L}_l^n$
- I_j contains (o, a) iff $S(R_j(o), Extent(C))$ is true.

S is a *scaling* operator, the most used are:

- $S_\exists(R_j(o), Extent(C))$ is true iff $R_j(o) \cap Extent(C) \neq \emptyset$.
- $S_{\forall\exists}(R_j(o), Extent(C))$ is true iff $R_j(o) \subseteq Extent(C) \wedge \exists x \in R_j(o), x \in Extent(C)$

Scaling operators

Operator	Attribute form	Condition
Universal (narrow)	$\forall\ r : c$	$r(o) \subseteq Ext(c)$
Covers	$\supseteq\ r : c$	$r(o) \supseteq Ext(c)$
Existential (wide)	$\exists\ r : c$	$r(o) \cap Ext(c) \neq \emptyset$
Universal strict	$\forall\exists\ r : c$	$r(o) \subseteq Ext(c)$ *and* $r(o) \neq \emptyset$
Qualified cardinality restriction	$\geq\ n\ r : c$	$r(o) \subseteq Ext(c)$ *and* $\lvert r(o) \rvert \geq\ n$
Cardinality restriction	$\geq\ n\ r : \top_{\mathcal{L}}$	$\lvert r(o) \rvert \geq\ n$

Relational scaling

Some properties of relational scaling:

- The homogeneity of concept descriptions is kept: all attributes are considered as binary (even relational attributes).
- Standard algorithms for building concept lattices can be directly reused.

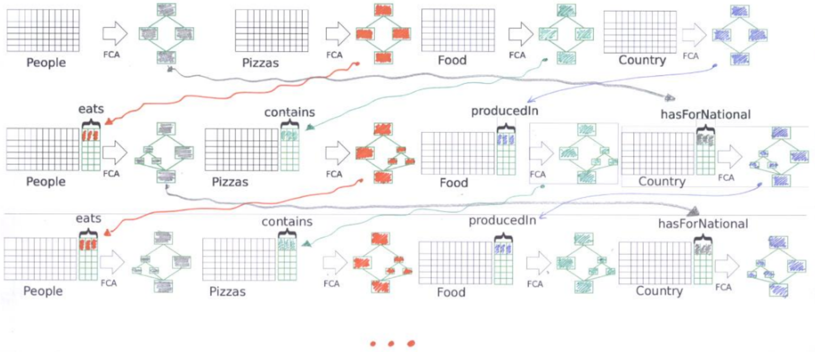
Relational scaling

After the relational scaling, the scaled relations are concatenated to the appropriate object-attribute relation (same object domain). This forms new relations (one per object set) which describe the objects by their classical attributes and their relational attributes. Concept lattices are built using these relations.

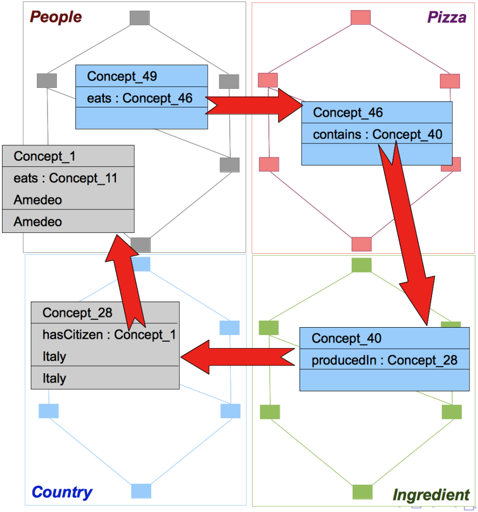
	producedIn : Concept_27	producedIn : Concept_28	producedIn : Concept_30	producedIn : Concept_31
basilic	x	x		
chicken	x		x	
cream	x			x
ham	x	x		
mozzarella	x	x		
mushroom	x		x	
olive	x	x		
tomato	x	x		
salmon	x		x	
swisscheese	x			x

Outlining the iterative algorithm

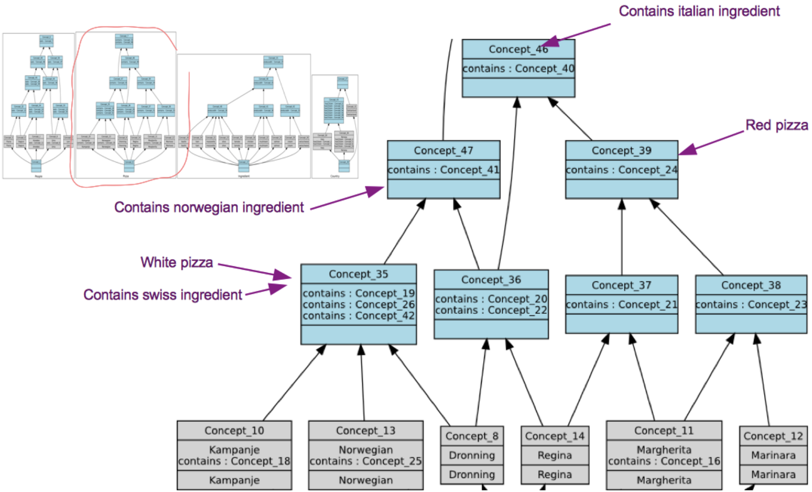
Learned concepts are used in the next steps to learn more



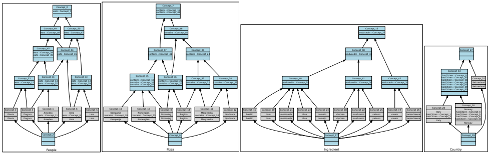
RCA view on data: A connected set of lattices



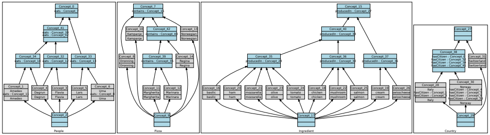
Reading classifications and implications through links



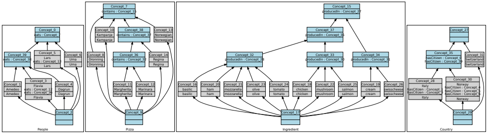
Alternative views by varying the scaling operators



$\exists eats \exists contains \exists producedIn \exists hasCitizen$



$\exists eats \forall contains \exists producedIn \exists hasCitizen$



$\forall eats \forall contains \exists producedIn \exists hasCitizen$

RCA engineering

Applications

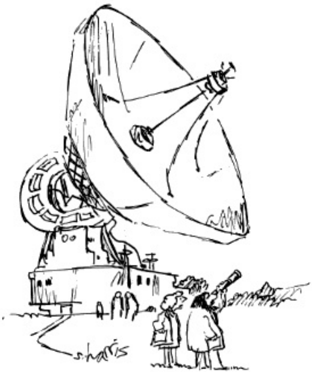
- Software engineering including: class model constructions, use case model reengineering, model transformations learning, analyzing and fixing bad smells in software, component architecture extraction, Web services classification ...
- Artificial Intelligence including: ontology constructions, data mining

Making RCA practical

- We can play with the scaling operators
- On specific (small parts of the) contexts, we can add queries
- We can apply RCA on part of the RCF, and stop at a given step
- We can observe the growth of (inferred) knowledge as an indicator about data

Outline

Relational queries

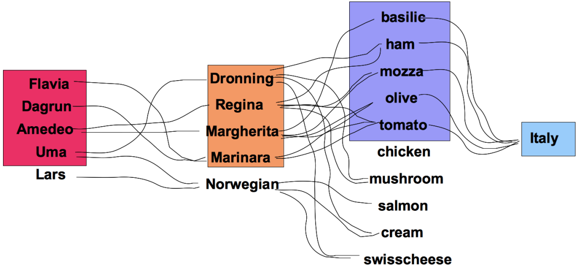


"Just checking."

http://astronomy.swin.edu.au/ gmackie/humour.html

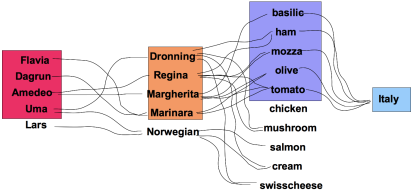
Relational queries

Consider this query: *We search for people, pizzas, ingredients and a country s.t. the country is Italy, people eat the pizzas, which contain ingredients produced in the country*



The answers are the tuples we can read from the schema, e.g.
< Flavia, Marinara, Olive, Italy >
< Dagrun, Marinara, Olive, Italy >
< Amedeo, Regina, Mozza, Italy > (etc.)

Relational queries



With FCA/RCA, we may add two interesting views:

- **classify the answers** (group answers concerning Flavia and Dagrun because they are based on the marinara pizza)
- **classify all objects to understand their relations w.r.t. the query**
  - Lars' preferences are not very far away from those of Uma, both like the Norwegian pizza; besides the Dronning pizza is not very different from the Norwegian pizza
  - most of the pizzas which are answers contain tomato

Relational queries

Approach

- Expressing the relational query to navigate the lattices
- A companion guiding algorithm with user interaction

Two main implementations

- By looking in each lattice for the searched relational attributes
- By classifying variables of the query

Relational query

Unary predicates associated with a formal context
We associate with a formal context K several unary predicates
P_K: to express that an object o belongs to the formal object set O
P_a: to express that an object o owns a specific attribute $a \in A$

E.g. $P_{K_{People}} P_{Italy}$

Binary predicates associated with a Relational Context Family
Let us consider a RCF $(\mathbb{K}, R)$.
To every r_{ij} relation in R, we associate a binary predicate
$P_{r_{ij}}$: to express that a pair of objects is connected via the relation

E.g. P_{eats}

Relational query

Definition (Relational query)

Let $\mathcal{V}$ denote a finite set of variables. A *relational query atom* is:

- either an expression $P(v_1)$ where v_1 is a variable from $\mathcal{V}$ and P a unary predicate associated with a formal context in $\mathbb{K}$,
- or $P_{r_{ij}}(v_1, v_2)$, where v_1 and v_2 are two variables from $\mathcal{V}$ or a variable and an object of O_i, $i \in \{1...n\}$, and $P_{r_{ij}}$ is a binary predicate associated with a relation r_{ij} in R.

A *relational query* with variables $\overrightarrow{v}$ is an expression $\varphi(\overrightarrow{v})$, which is a conjunction of relational query atoms involving variables of $\mathcal{V}$ and objects in O_i, where $i \in \{1...n\}$.

Relational query

We search for people, pizzas, ingredients and a country s.t. the country is Italy, people eat the pizzas, which contain ingredients produced in the country

$$P_{K_{People}}(q_{people}) \wedge P_{K_{Pizza}}(q_{pizza}) \wedge P_{K_{Ingredient}}(q_{ingredient}) \wedge P_{K_{Country}}(q_{country}) \wedge P_{Italy}(q_{country}) \wedge P_{eats}(q_{people}, q_{pizza}) \wedge P_{contains}(q_{pizza}, q_{ingredient}) \wedge P_{producedIn}(q_{ingredient}, q_{country})$$

Classifying variables of the query

An example of a modified object-attribute context

Variables are added as new objects, here q-person is added in K_{People}.
It may have attributes depending from the query.

I_{People}	**Amedeo**	**Flavia**	**Dagrun**	**Lars**	**Uma**
Amedeo	×				
Flavia		×			
Dagrun			×		
Lars				×	
Uma					×
q-person					

Classifying variables of the query

An example of a modified object-object context

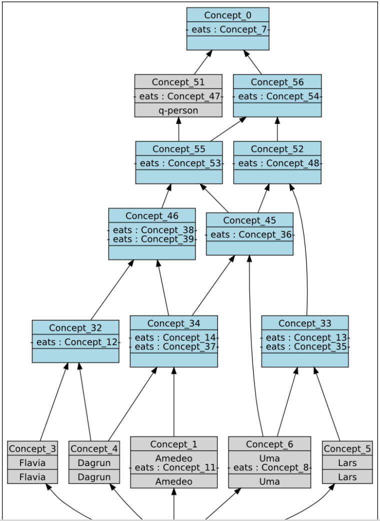
We look for people (q-people) connected to pizzas (q-pizzas)

	Dronning	Kampanje	Margherita	Marinara	Norwegian	Regina	**q-pizza**
Amedeo			×			×	
Flavia				×			
Dagrun				×		×	
Lars					×		
Uma	×				×		
q-person							**X**

Reading answers by concept lattice family navigation

Answers are below the concept introducing q-people.

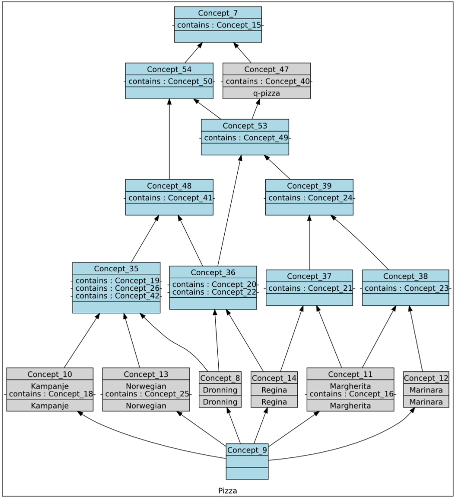
We see patterns in answers (e.g. similar consumer profiles Lars-Uma; Flavia-Dagrun)



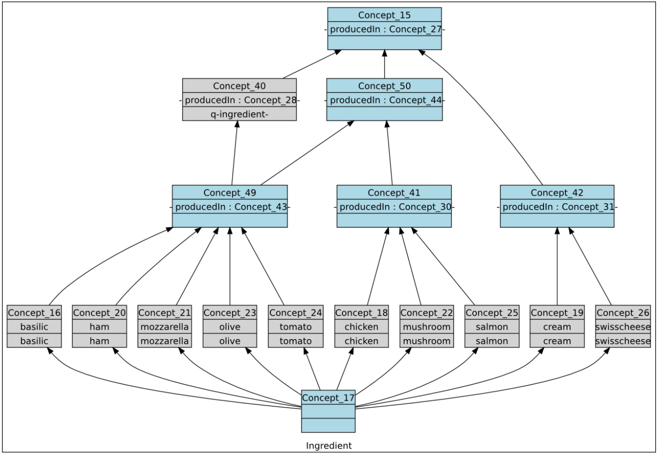
Reading answers by concept lattice family navigation

Answers for pizzas are below the concept introducing q-pizza.

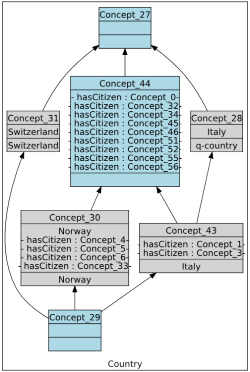
Concept_54 has been introduced to factorize something that all initial objects (but not the object associated with the query) have: containing ingredients produced in a country which has citizens in the studied context.



Reading answers by concept lattice family navigation



Reading answers by concept lattice family navigation



Relational queries

FCA/RCA is interesting for discovering unanticipated knowledge

Why/When considering queries?

- to understand how existing objects correspond / don't correspond to the query
- to discover unanticipated knowledge about answers and the other objects

Discussion

- When the query has circuits, define a navigation order
- Defining queries with variants (using other scaling operators)
- Based on a navigation algorithm, propose a tool to help the user navigating

Specific relational schemes

In a sense, when we choose a RCF, we choose a relational scheme allowing to search data with a specific idea in mind.

Relation	Step 1	Step 2	Step 3	...
eats	$\forall r : c$	$\exists r : c$	$\exists r : c$	$\exists r : c$
contains	$\geq n\, r : c$	$\forall r : c$	$\exists r : c$	$\exists r : c$
producedIn	$\geq n\, r : c$	$\forall r : c$	$\exists r : c$	$\exists r : c$
hasForNational	$\exists r : c$	$\forall r : c$	$\exists r : c$	$\exists r : c$
...	...	...	...	

To discover for example:

- People that eat only pizza containing at least n ingredients produced in Norway
- Countries where at least one national eats at least one pizza that contains at least one food produced in Italy

Note: at one step, several scaling operators can be applied to the same relational context (giving several scaled contexts based on a same relational context)

Outline

Erratic RCA

http://www.cafepress.com/+labyrinth+greeting_cards

Variations on the RCA algorithm

- to focus step-by-step on part of the data
- for scalability
- to launch the process in specific cases

An example for giving some insight
See Xavier Dolques' talk for more details

Strange animals

I_{Motion}	
fly	
walk	

I_{Animal}	
flamingo	
flying squirrel	
geek	

I_{Places}	
lake	
forest	
city	

I_{Food}	
artemia	
lichen	
pizza	

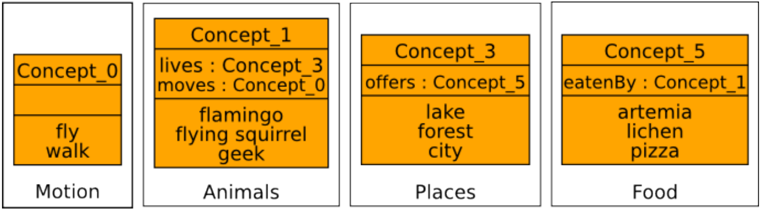
I_{lives}	lake	forest	city
flamingo	×		
flying squirrel		×	
geek			×

I_{moves}	fly	walk
flamingo	×	
flying squirrel	×	
geek		×

I_{offers}	artemia	lichen	pizza
lake	×		
forest		×	
city			×

$I_{eatenBy}$	flamingo	flying squirrel	geek
artemia	×		
lichen		×	
pizza			×

Classical RCA algorithm



For the rest of the example, these lattices are denoted by TMotion_T1form, TAnimals_T1form, TPlaces_T1form, TFood_T1form

Erratic RCA

An example of approach

- Successively choose a set of objects, and introduce the relations this set of objects is a domain
- When no lattice exists on a set of objects: introduce relations **without** relational scaling
- When a lattice exists on a set of objects: introduce relations **with** relational scaling

Erratic RCA

Introduce relation without relational scaling

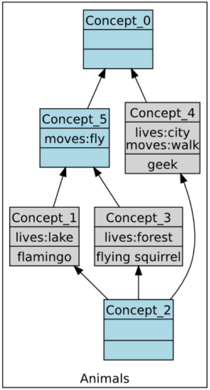
$I_{Animals} + I_{lives} + I_{moves}$	lives:lake	lives:forest	lives:city	moves:fly	moves:walk
flamingo	×			×	
flying squirrel		×		×	
geek			×		×

We will denote this kind of data by:
Animals+lives(Places)+moves(Motions)

Erratic RCA

Building a lattice at a step, ex. 1

$I_{Animals} + I_{lives} + I_{moves}$	lives:lake	lives:forest	lives:city	moves:fly	moves:walk
flamingo	×			×	
flying squirrel		×		×	
geek			×		×



We denote this step by:
Animals+lives(Places)+moves(Motions) –> TAnimals_T6form

Erratic RCA

Introduce relation with relational scaling

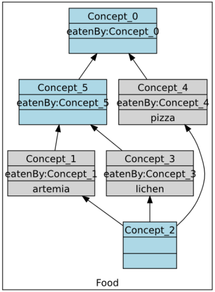
$I_{Food} + I_{eatenBy(TAnimals\_T6form)}$	∃ eatenBy:Concept_0	∃ eatenBy:Concept_1	∃ eatenBy:Concept_3	∃ eatenBy:Concept_4	∃ eatenBy:Concept_5	
artemia	×	×			×	
lichen	×		×		×	
pizza	×			×		

We will denote this kind of data by:
Food+eatenBy(TAnimals_T6form)

Erratic RCA

Building a lattice at a step, ex. 2

$I_{Food} + I_{eatenBy(TAnimals\_T6form)}$	∃ eatenBy:Concept_0	∃ eatenBy:Concept_1	∃ eatenBy:Concept_3	∃ eatenBy:Concept_4
artemia	×	×		
lichen	×		×	
pizza	×			×

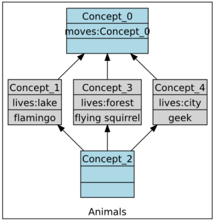


We denote this step by:
Food+eatenBy(TAnimals_T6form) –> TFood_T6form

Erratic RCA

Alternatively, using the lattice Motion reduced to a concept, ex. 3

$I_{Animals} + I_{lives} + I_{moves(TMotion)}$	lives:lake	lives:forest	lives:city	∃ moves:Concept_0
flamingo	×			×
flying squirrel		×		×
geek			×	×



We denote this step by:
Animals+lives(Places)+moves(TMotion_T1form) –> TAnimals_T3form

Erratic RCA

1. Animals+lives(Places)+moves(motion)–> TAnimals_T6form
2. Places+offers(Food) –> TPlaces_T3form
3. Food+eatenBy(TAnimals_T6form) –> TFood_T6form
4. Motion –> TMotion_T1form
5. Animals+lives(TPlaces_T3form)+moves(Tmotion_T1form)–> TAnimals _T3form
6. Places+offers(TFood_T6form) –> TPlaces_T6form
7. Food+eatenBy(TAnimals_T3form) –> TFood_T3form
8. Animals+lives(TPlaces_T6form)+moves(Tmotion_T1form)–> TAnimals_T6form
9. Places+offers(TFood_T3form) –> TPlaces_T3form
10. ...

A possibly diverging path!

Erratic RCA

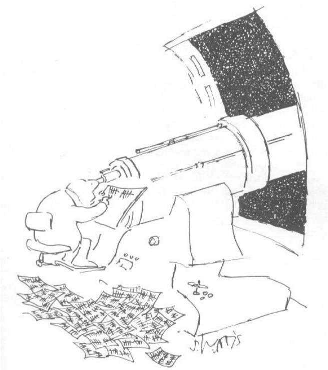
Lessons we can draw:

- concepts emerge step-by-step, only a partial view is given, which is interesting in an interactive process
- we may converge more rapidly (on other examples)
- we have to pay attention to possible on non-monotonic lattice construction

Applications, interpretation of the lattices and perspectives: see Xavier's Talk

Outline

Observing the concept lattice growth in RCA



http://astronomy.swin.edu.au/ gmackie/humour.html

Observing the concept lattice growth in RCA

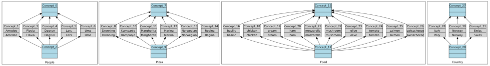
- to understand how data are connected and concept formation propagates
- applied to evolving datasets, to understand evolution of the concept formation
- to pre-cluster data before RCA

work inspired by experiments in IS applications done with: B. Amar, A. Osman Guedi, A. Miralles, C. Nebut, T. Libourel

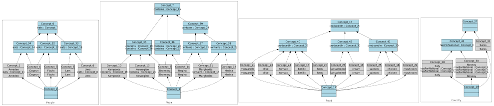
Concept lattice family evolution

On the pizza example, successive steps show the growth of conceptual knowledge

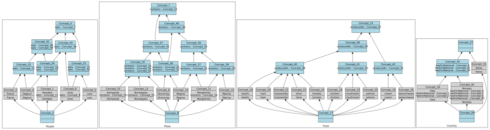
step 0.



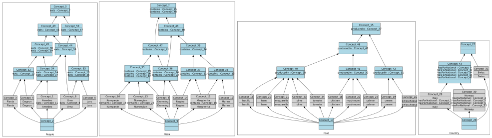
step 1.



step 2.



step 3.



Pre-clustering a dataset for scalability

An example of a unique dataset (object-attribute context)

		1	2	3	4
z1		×			
z2		×			
z3		×			
a1			×		
a2			×		
a3			×		
b1				×	
b2				×	
b3				×	
c1					×
c2					×
c3					×
c4					×
c5					×
c6					×
d1	×	×			
d2	×		×		
d3	×			×	

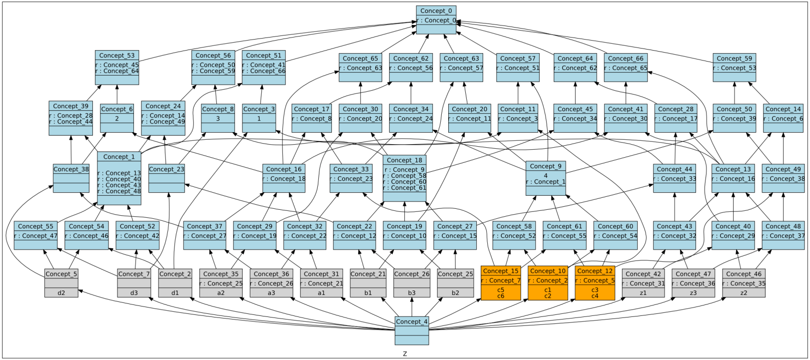
Pre-clustering a dataset for scalability

An example of a unique dataset (object-object context)

	z1	z2	z3	a1	a2	a3	b1	b2	b3	c1	c2	c3	c4	c5	c6	d1	d2	d3
z1				×														
z2					×													
z3						×												
a1							×											
a2								×										
a3									×									
b1										×		×						
b2											×			×				
b3													×		×			
c1																×		
c2																×		
c3																	×	
c4																	×	
c5																		×
c6																		×
d1	×	×																
d2		×	×															
d3	×		×															

Pre-clustering a dataset for scalability

The concept lattice family (one lattice)



Pre-clustering a dataset for scalability

Alternative view on same data, by dividing the dataset and the links

	1
z1	×
z2	×
z3	×

	2
a1	×
a2	×
a3	×

	3
b1	×
b2	×
b3	×

	4
c1	×
c2	×
c3	×
c4	×
c5	×
c6	×

	1	2	3	4
d1	×	×		
d2	×		×	
d3	×			×

I_{r1}	a1	a2	a3
z1	×		
z2		×	
z3			×

I_{r2}	b1	b2	b3
a1	×		
a2		×	
a3			×

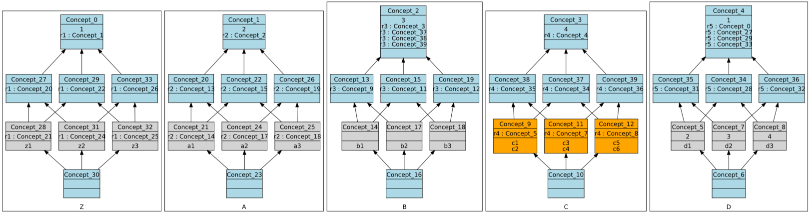
I_{r3}	c1	c2	c3	c4	c5	c6
b1	×		×			
b2		×			×	
b3				×		×

I_{r4}	d1	d2	d3
c1	×		
c2	×		
c3		×	
c4		×	
c5			×
c6			×

I_{r5}	z1	z2	z3
d1	×	×	
d2		×	×
d3	×		×

Pre-clustering a dataset for scalability

The obtained concept lattice family



Outline

Conclusion / Perspectives

Conclusion

- RCA gives a view of data via a set of connected lattices
- Past and ongoing applications on various data

Perspectives

- RCA engineering
  - Extracting classifications and implications
  - Queries and RCA
  - Understanding concept lattices growth
  - Erratic RCA
  - What about Metrics: e.g. a version of stability for RCA?