ANR-17-CE23-0007

ELKER

Enhancing link keys: extraction and reasoning
Étendre les clés de liage: extraction et raisonnement

D1.2 Relational concept analysis for circular link key extraction

---

Coordinator: Jérôme Euzenat
With contributions from: Manuel Atencia, Jérôme David, Jérôme Euzenat, Amedeo Napoli, Jérémy Vizzini

Quality reviewer:	Specify quality controller
Reference:	ELKER/D1.2/v1.0
Project:	ELKER ANR-17-CE23-0007
Date:	December 9, 2021
Version:	1.0
State:	final
Destination:	public

EXECUTIVE SUMMARY

Linked data aims at publishing data expressed in RDF (Resource Description Framework) at the scale of the worldwide web. These heterogeneous datasets interoperate by publishing links which identify individuals across them.

Such links may be found by using a generalisation of keys in databases, called link keys, which apply across datasets. They specify the pairs of properties to compare for linking individuals belonging to different classes of the datasets.

Here, we extend the work presented in Deliverable 1.1 by dealing with circular dependencies between concepts and thus link keys.

For that purpose, we use relational concept analysis (RCA), an extension of FCA taking relations between concepts into account. This yields an elegant formulation of extracting link keys in presence of circular dependencies (§3). We show that it enables to extract the optimal link keys even in the presence of cyclic dependencies. Moreover, the proposed process does not require information about the alignments of the ontologies to find out from which pairs of classes to extract link keys.

However, further investigation show that the RCA process, as it is currently defined may fall short in finding useful link keys. In practice, this observation is not really a problem to RCA because in most data sets, there are explicit data able to ground a solution. However, from a theoretical perspective such problems are worth investigating.

To illustrate this problem, we define a ultra-simplified version of RCA that we call RCA^0. RCA^0 concentrates on dependencies between formal contexts. Introducting RCA^0 has several advantages: it provides examples in which the raised problems are clearly apparent; it thus allows to analyse directly what the sources of such problems are; and finally it makes the presentation of the results and solutions more simple.

To pinpoint the source of the problem, we provide examples of RCA shortcomming in a collection of examples using classical RCA or the link key extraction formulation (§4).

We redefine the semantics of RCA with respect to the notion of closed concepts (§5). A solution to an RCA problem is characterised as a fixed-point of a specific function called expansion function. This requires to precisely define the semantics of scaling which raises problems in case of circularity. We show that the previously available semantics, which we call well-founded semantics, only computes the least fixed-point. This semantics is extended by introducing the notion of self-supported lattices, whose attributes only refer to concepts of this lattice. Interesting lattices for RCA are self-supported fixed points.

We consider alternative to the RCA algorithm to find other solutions (§6). In particular, we develop an algorithm complementary to the RCA algorithm. It starts with the power set lattice of the set of objects and filters out non-self supported concepts. It has the drawback of starting with a large lattice. We show that it computes the greatest fixed-point. We also discuss algorithms and hints to generate all valid lattices for RCA^0.

This work leaves currently some questions open: (*i*) Does this apply to more complex setting than RCA^0? (*ii*) How to apply this to link key extraction.

Chapter 3 of this report have been published in [Atencia et al. 2020]. Chapter 4 and 5 are extensions of [Euzenat 2021]. The rest is unpublished.

DOCUMENT INFORMATION

Project number	ANR-17-CE23-0007	**Acronym**	ELKER
Full Title	Enhancing link keys: extraction and reasoning Étendre les clés de liage: extraction et raisonnement		
Project URL	`https://project.inria.fr/elker/`		
Document URL			

Deliverable	**Number**	1.2	**Title**	Relational concept analysis for circular link key extraction
Work Package	**Number**	1	**Title**	Link key extraction

Date of Delivery	**Contractual**	M36	**Actual**	23/07/2021
Status	final		final ⊠	
Nature	prototype □ report ⊠ dissemination □			
Dissemination level	public ⊠ consortium □			

Authors (Partner)	Manuel Atencia, Jérôme David, Jérôme Euzenat, Amedeo Napoli, Jérémy Vizzini			
Resp. Author	**Name**	Jérôme Euzenat	**E-mail**	Jerome.Euzenat@inria.fr
	Partner	LIG		

Abstract **(for dissemination)**	A link key extraction procedure in case of circular dependencies is presented. It uses relational concept analysis and extends the procedure of Deliverable 1.1. This leads to investigate more closely the semantics of relational concept analysis which is given in terms of fixed points. Extracting all fixed points may offer more link key candidates to consider.
Keywords	Formal Concept Analysis, Relational Concept Analysis, Linked data, Link key, Data interlinking, Resource Description Framework

Version Log			
Issue Date	**Rev No.**	**Author**	**Change**
03/05/2019	1	J. Euzenat	Initial version
09/05/2019	2	J. Euzenat	Inclusion of RCA material from the DAM paper
19/11/2020	3	J. Euzenat	Revision of the content
02/12/2020	4	J. Euzenat	Inclusion of material from ongoing fixed-point paper
08/12/2020	5	J. Euzenat	Reorganised the outline
18/04/2021	6	J. Euzenat	Included content of ICFCA paper (and extra)
05/05/2021	7	J. Euzenat	Developed $Q^{\infty} \circ E^{\infty}$; cleaned up the paper
19/05/2021	8	J. Euzenat	Revamped the whole paper; finished gfp algorithm
14/07/2021	9	J. Euzenat	Finished the full algorithm; revised the whole paper
23/07/2021	10	J. Euzenat	Completed all missing parts
09/12/2021	11	J. Euzenat	Taken into account quality control remarks

TABLE OF CONTENTS

1. Introduction

Linked data aims at publishing data expressed in RDF (Resource Description Framework) at the scale of the worldwide web [Bizer et al. 2009; Heath and Bizer 2011]. These datasets interoperate through links which identify individuals across heterogeneous datasets. Data interlinking, the problem of linking pairs of nodes corresponding to the same resource in RDF graphs, is an important task for linked open data.

Different approaches and methods have been proposed to address the problem of automatic data interlinking [Ferrara et al. 2011; Nentwig et al. 2017]. Most of them are based on numerical methods that measure a similarity between entities and consider that the closest the entities, the more likely they are the same [Volz et al. 2009; Ngonga Ngomo and Auer 2011]. Other work takes a logical approach to data interlinking and can leverage reasoning methods [Saïs et al. 2007; Al-Bakri et al. 2015; Hogan et al. 2012].

We introduced the notion of *link keys* as a way to identify such node pairs [Euzenat and Shvaiko 2013; Atencia et al. 2014]. Link keys generalise keys in relational algebra in three ways: (1) they apply across two data sets instead of a single one, (2) they take into account multiple values for the same attribute, and (3) attribute values may be other objects. The latter makes link keys eventually dependent on each others.

Link keys specify the pairs of properties to compare for linking individuals belonging to different classes of the datasets. An example of a link key is:

$$\{\langle \mathsf{auteur}, \mathsf{creator}\rangle\}\{\langle \mathsf{titre}, \mathsf{title}\rangle\}\ \mathit{linkkey}\ \langle \mathsf{Livre}, \mathsf{Book}\rangle$$

stating that whenever an instance of the class Livre has the same values for property auteur as an instance of class Book has for property creator and they share at least one value for their property titre and title, then they denote the same entity.

Clearly, such a link key may depend on another one as, for instance, properties auteur and creator have values in the Écrivain and Writer classes respectively. Identifying their values will then resort to another link key:

$$\{\langle \mathsf{prénom}, \mathsf{firstname}\rangle\}\{\langle \mathsf{nom}, \mathsf{lastname}\rangle\}\ \mathit{linkkey}\ \langle \mathsf{Écrivain}, \mathsf{Writer}\rangle$$

This situation may be rendered even more intricate if Écrivain and Writer were instead identified from the values of their properties ouvrages and hasWritten refering to instances of Livre and Book. We would then face interdependent link keys.

We have already proposed an algorithm for extracting some types of link keys [Atencia et al. 2014]. This method may be decomposed in two distinct steps: (1) identifying link key candidates, followed by (2) selecting the best link key candidates according to quality measures. In Deliverable 1.1, we have shown how to encode the functional link key extraction problem in relational databases into Formal concept analysis so that candidate link keys correspond to formal concepts [Atencia et al. 2014]. Formal Concept Analysis (FCA [Ganter and Wille 1999]) is a useful tool for inducing a classification structure from data.

Relational Concept Analysis (RCA [Rouane-Hacene et al. 2013a]) is one of its extensions allowing to take advantage of relationships between objects to extract dependent concept lattices. One of its strong point is its ability to deal with circular dependencies between objects. However, RCA aims at identifying and characterising concepts within data, though our goal is to identify link keys across two datasets. Hence, the framework has to be adapted to this end with specific scaling operators.

In this report, we show how relational concept analysis can be used to deal with circular dependencies and hence to extract directly families of interdependent link key candidates from RDF data sets [Atencia et al. 2020]. This method generalises directly the one presented for non dependent link keys [Atencia et al. 2014] and link keys over the relational model [Atencia et al. 2014]. In this context,

the concepts of extracted lattices are link key candidates which will be selected on the basis of two independent measures [Atencia et al. 2014].

Although the result returned by RCA is solid and useful, it may not be the only possible result. The relational structure, when containing circuits, has the capability to induce richer lattice structures. Indeed, in the absence of information or of reason to separate objects, RCA classifies them within the same concept. On the contrary, in the absence of information or of reason to aggregate objects, it is possible to assign them to different concepts, though generating stable concept lattices. A good compromise may sometimes reside in between these two extremes. As a data mining procedure, RCA can be useful in returning all possible structures and not necessarily the safest ones.

This problem is not specific to link key extraction but apply to RCA in general. However, in the target RCA application, extracting the core classes of a description logic ontology, this may not be critical. On the contrary, for the extraction of link key candidates which will be evaluated to find the best one, not having them all may be a problem. As a data mining task, RCA is more useful if it generates all the possible link key candidates.

Hereafter, we illustrate the considered problem on RCA^0, a minimal version of RCA. Although RCA^0 is simply a convenient way to illustrate the problem it requires solutions that will apply to RCA as a whole.

Understanding the nature of the problem and its relation with RCA leads to consider its semantics. The standard semantics of RCA [Rouane-Hacene et al. 2013b] focusses on the grounding of the process. We redefine this semantics on properties directly characterising the solutions.

We first consider the core function involved in the classical RCA algorithm and identify acceptable results as the fixed points of this function. We show, in the case of RCA^0, that the classical RCA semantics corresponds to extracting its least fixed point.

We also provide a direct way to generate the greatest fixed point. However, although RCA extracts the minimal fixed point in its simplest form, this is not the case of the greatest fixed point: it would make reference to non-existent concepts. Hence we introduce the notion of self-supported concept lattice, so that the acceptable RCA results would be self-supported fixed points.

Outline The outline of the report is as follows. Related work has been addressed in Deliverable 1.1 and Deliverable 1.3, hence we first introduce the problem and notations used in the paper as well as the basics of formal and relational concept analysis (§2). This allows us to generalise the notion of scaling that will be useful later. Then, we provide the RCA encoding of the problem of cyclic dependent link key candidate extraction through the definition of a relational context and specific scaling operators (§3). All this ends the coverage of [Atencia et al. 2020]. Then we provide examples of problems raised by the solutions provided by RCA (§4). In order to explain these examples, we provide a fixed-point semantics for RCA based on a context-expansion function which allows to characterise the classical RCA semantics (§5). However, this semantics does not fully characterise interesting lattices, so we introduce the complementary notion of self-supported concept lattices. These self-supported fixed points are those lattices that can be returned by an extended RCA. Finally, we provide algorithms able to extract the greatest self-supported fixed-point and the set of all self-supported fixed points in this restriction of RCA (§6).

Acknowledgements

We thank Petko Valtchev for comments and suggestion on an earlier version of this work. We thank Philippe Besnard for pointing to the Knaster-Tarski theorem.

2. Preliminaries

We mix preliminaries with related works for reasons of space, but also because the paper directly builds on this related work.

Table 2.1 provides a list of symbols used in this deliverable.

2.1 Basics of formal concept analysis

Formal Concept Analysis (FCA) [Ganter and Wille 1999] starts with a binary context $\langle G, M, I \rangle$ where G denotes a set of objects, M a set of attributes, and $I \subseteq G \times M$ a binary relation between G and M, called the incidence relation. The statement gIm is interpreted as "object g has attribute m". Two operators $\cdot^{\uparrow}$ and $\cdot^{\downarrow}$ define a Galois connection between the powersets $\langle 2^G, \subseteq \rangle$ and $\langle 2^M, \subseteq \rangle$, with $A \subseteq G$ and $B \subseteq M$:

$$A^{\uparrow} = \{m \in M \mid gIm \text{ for all } g \in A\}$$
$$B^{\downarrow} = \{g \in G \mid gIm \text{ for all } m \in B\}$$

The operators $\cdot^{\uparrow}$ and $\cdot^{\downarrow}$ are decreasing, i.e. if $A_1 \subseteq A_2$ then $A_2^{\uparrow} \subseteq A_1^{\uparrow}$ and if $B_1 \subseteq B_2$ then $B_2^{\downarrow} \subseteq B_1^{\downarrow}$. Intuitively, the less objects there are, the more attributes they share, and dually, the less attributes there are, the more objects have these attributes. It can be checked that $A \subseteq A^{\uparrow\downarrow}$ and that $B \subseteq B^{\downarrow\uparrow}$, that $A^{\uparrow} = A^{\uparrow\downarrow\uparrow}$ and that $B^{\downarrow} = B^{\downarrow\uparrow\downarrow}$.

For $A \subseteq G$, $B \subseteq M$, a pair $\langle A, B \rangle$, such that $A^{\uparrow} = B$ and $B^{\downarrow} = A$, is called a formal concept, where A is the extent and B the intent of $\langle A, B \rangle$. Moreover, for a formal concept $\langle A, B \rangle$, A and B are closed sets for the closure operators $\cdot^{\uparrow\downarrow}$ and $\cdot^{\downarrow\uparrow}$, respectively, i.e. $A^{\uparrow\downarrow} = A$ and $B^{\downarrow\uparrow} = B$.

Concepts are partially ordered by $\langle A_1, B_1 \rangle \leq \langle A_2, B_2 \rangle \Leftrightarrow A_1 \subseteq A_2$ or equivalently $B_2 \subseteq B_1$. With respect to this partial order, the set of all formal concepts is a complete lattice called the concept lattice of $\langle G, M, I \rangle$.

Formal concept analysis can be considered as a function that associates to a formal context $\langle G, M, I \rangle$ its concept lattice $\langle C, \leq \rangle = FCA(\langle G, M, I \rangle)$ (or $\underline{\mathfrak{B}}(G, M, I)$ [Ganter and Wille 1999]). By abuse of language, when a variable L denotes a concept lattice $\langle C, \leq \rangle$, L will also be used to denote C.

2.2 Extending formal concept analysis

Formal concept analysis is defined on relatively simple structures hence many extensions of it have been designed. The may allow FCA to (a) deal with more complex input structure, and/or (b) generate more expressive and interpretable knowledge structures.

2.2.1 Scaling: a generalisation

Scaling is one type of extension of type (a). A scaling operation $\varsigma : \mathcal{X} \mapsto 2^D$ generates boolean attributes named after a language D from a structure $\Sigma \in \mathcal{X}$. In scaled contexts, this language can be interpreted so that the incidence relation I is immediately derived from the attribute m following:

$$\Sigma \models gIm$$

In FCA, $D = M$ and I is provided by its matrix:

$$K \models gIm \text{ iff } \langle m, g \rangle \in I$$

Hence, adding attributes to a context under such a structure may be performed as:

$$K^{\Sigma}_{+M'}(\langle G, M, I \rangle) = \langle G, M \cup M', I \cup \{\langle g, m \rangle \in G \times M' \mid \Sigma \models m(g)\} \rangle$$

G	set of objects ($g \in G$)
M	set of attribute ($\subseteq D$, $m \in M$)
I	incidence relation ($\subseteq M \times G$)
J	'ternary' element used in conceptual scaling
R	set of relations ($\subseteq G \times G'$, $r \in R$)
K	formal contexts
Ω	set of scaling operations ($\varsigma \in \Omega$)
$N(G)$	concept names (given after extent, $\subseteq 2^G$)
D	property language for expressing attributes (inspired from Pattern structure)
L	concept lattice
C	set of formal concepts ($\subseteq 2^{G \times M}$, $c \in C$)
A	description logic ABox
T	description logic TBox
Σ	demantic structure grounding scaling

Table 2.1: Some symbols used in this document.

and suppressing them as:

$$K^{\Sigma}_{-M'}(\langle G, M, I\rangle) = \langle G, M \setminus M', I \setminus \{\langle g, m\rangle \in G \times M' \mid \Sigma \vDash m(g)\}\rangle$$

Applying a scaling operation ς to a formal context K following a structure Σ can be thus decomposed into (*i*) determining the set of attributes $\varsigma(\Sigma)$ to add, and (*ii*) extending the context with such attributes:

$$\sigma_{\varsigma}(K, \Sigma) = K^{\Sigma}_{+\varsigma(\Sigma)}(K)$$

This unified view of scaling may be applied to many available scaling operators. We discuss these below.

2.2.2 Conceptual scaling

Attributes found in data sets typically do not range in Booleans, but instead in numbers, intervals, strings. Such data can be represented as a many-valued context $\Sigma = \langle G, M, W, J\rangle$, where G is a set of objects, M a set of attributes, W a set of values, and J a ternary relation defined on the Cartesian product $G \times M \times W$. The fact $\langle g, m, w\rangle \in J$ or simply $m(g) = w$ means that object g takes the value w for the attribute m. In addition, when $\langle g, m, w\rangle \in J$ and $\langle g, m, v\rangle \in J$ then $w = v$ [Ganter and Wille 1999]: in FCA, "many-valued" means that the range of an attribute may include more than two values, but for any object, the attribute can only have one of these values.

Conceptual scaling can be used for transforming such a many-valued context into a one-valued context. For example, if $W_m = \{w_1, w_2, \ldots w_p\} \subseteq W$ denotes the range of the attribute m, then a scale of elements "$m = w_i, \forall w_i \in W_m$" is used for binarising the initial many-valued context. Intuitively, a scale splits the range W_m of a many-valued attribute m into a set of p binary attributes "$m = w_i, i = 1, \ldots, p$". There are many possible scalings and some of them are detailed in [Ganter and Wille 1999] (see Table 2.2).

For instance, for nominal scaling, given a set W of values and a set N of properties taking these values, we can define $D^{=}_{N,W} = \{n = w | n \in N \text{ and } w \in W\}$. Then, given a many-valued context $\Sigma = \langle G, N, W, J\rangle$. Attributes of N are nominaly scaled on $D^{=}_{N,W}$ so that, $\forall n \in N$:

$$\langle G, N, W, J\rangle \vDash gI(n = w) \text{ iff } \langle g, n, w\rangle \in J$$

or, more precisely, it is scaled through a nominal scale S which is the relational context corresponding to the equality relation on W.

name	language (D)	scale (S)	Σ	condition ($m(g)$)
FCA	m	-	I	$m(g)$
dichotomic	$n = v$	monocolumn	$\langle G,N,W,J\rangle$	$n(g) = v$
nominal	$n = w$	diagonal	$\langle G,N,W,J\rangle$	$n(g) = w\ \forall w \in W$
ordinal	$n \leq w$	triangular	$\langle G,N,W,J\rangle$	$n(g) \leq w\ \forall w \in W$
inter-ordinal	$n \leq w$, $n \geq w$	-	$\langle G,N,W,J\rangle$	$n(g) \leq w$ or $n(g) \geq w\ \forall w \in W$
contranominal	$n \neq w$	antidiagonal	$\langle G,N,W,J\rangle$	$n(g) \neq w\ \forall w \in W$

Table 2.2: Conceptual scaling operators (inspired from [Ganter and Wille 1999]).

The same can be built for other type of scaling, e.g.

$$\langle G,N,W,J\rangle \vDash gI(n\ \leq w) \text{ iff } \langle g,n,v\rangle \in J \wedge v \leq w$$

for ordinal scaling.

These scaling operations only use a simple structure, i.e. $\Sigma = \langle G,M,W,J\rangle$ in which everything is stored in J and D is expressed as predicates, e.g. $\cdot = v$ for nominal scaling or $\cdot \leq n$ for ordinal scaling. $\vDash$ is the evaluation of the predicate for the value, hence they can be called structural scaling.

2.2.3 Relational scaling

Relational scaling operations (σ_ς) considered in [Rouane-Hacene et al. 2013a] are based on a struture $\Sigma = \langle R,L\rangle$ made of a set of relations $R = \{r_y\}_{y\in Y}$, i.e. relations $r_y \subseteq G_x \times G_z$ between two sets of objects, and a family $L = \{L_x\}_{x\in X}$ of concept lattices whose extent is a subset of G_x.

Given the finite set of objects G_x from which each lattice is built, the set of concepts that can be created from such contexts is finite and moreover each concept can be identified by its extent. Hence, $N(G_x) = 2^{G_x}$ is the set of all concept names that may be used in any such concept lattice[1]; the extent of a so-named concept will be the set of objects in its name. The set of attributes that can be scaled from ς on r of codomain G_z is thus $D_{\varsigma,r,N(G_z)} = \{\varsigma r.c \mid c \in N(G_z)\}$.

For example, qualified existential scaling ($\exists$) adds attributes $\exists r.c$ for $r \in R$, $r \subseteq G_x \times G_z$, $c \in L_z$ and $\vDash$ checks that

$$\langle R,L\rangle \vDash gI\exists r.c \text{ iff } \exists g'; \langle g,g'\rangle \in r \wedge g' \in extent(c)$$

Various relational scaling operations are used in RCA, such as existential, strict and wide universal, min and max cardinality, which all follow the classical role restriction semantics of description logics [Baader et al. 2003] (see Table 2.3). The set of attributes obtained from relational scaling may be large but remains finite. Cardinality constraints may entail infinite sets of concepts in theory, but in practice the set of meaningful concepts are bound by $|G_z|$ which is finite.

In fact, RCA may be considered as a very general way to apply scaling across contexts. New operators may be provided [Braud et al. 2018; Wajnberg 2020], such as those that we used for extracting link keys [Atencia et al. 2020].

2.2.4 Logical scaling

Logical scaling [Prediger 1997] has been introduced for more versatile languages such as description logics and SQL. It introduces query results within formal contexts. In this case, Σ is a logical theory or database tables, D the set of formulas of the logic or queries (Q) and $\vDash$ is entailment or query evaluation.

$$\Sigma \vDash gIQ \text{ iff } \Sigma \vDash Q(g)$$

[1] A similar remark is made in [Wajnberg 2020, §4.1.2].

name	language (D)	scale (S)	Σ	condition ($m(g)$)
existential	$\exists r$		R	$r(g) \neq \varnothing$
universal (wide)	$\forall r.C$		R, L	$r(g) \subseteq extent(C)$
strict universal	$\forall \exists r.C$		R, L	$r(g) \neq \varnothing \wedge r(g) \subseteq extent(C)$
contains (wide)	$\forall C.r$		R, L	$extent(C) \subseteq r(g)$
strict contains	$\forall \exists C.r$		R, L	$extent(C) \neq \varnothing \wedge extent(C) \subseteq r(g)$
qualified existential	$\exists r.C$		R, L	$r(g) \cap extent(C) \neq \varnothing$
qualified min cardinality	$\leq_n r.C$		R, L	$\lvert r(g) \cap extent(C)\rvert \leq n$
qualified max cardinality	$\geq_n r.C$		R, L	$\lvert r(g) \cap extent(C)\rvert \geq n$
$\forall$-condition	$\forall \langle r, r' \rangle_k$		$R \times R', L_{C \times C'}$	$r(g) =_k r'(g')$
$\exists$-condition	$\exists \langle r, r' \rangle_k$		$R \times R', L_{C \times C'}$	$r(g) \cap_k r'(g') \neq \varnothing$

Table 2.3: Relational scaling operators (inspired from [Rouane-Hacene et al. 2013a; Braud et al. 2018]) and additional link key condition scaling operators operators (see Section 3.1).

Here, the nearly identical notation shows the relevance of this generalisation. We expressed it here with respect to one individual g, so it applies to unary queries or formulas with one variable placeholder. However, it is useful to generalise to contexts in which individuals in G are elements of the products of sets of individuals.

2.2.5 Relational scaling as logical scaling

The type of scaling used by RCA, relational scaling, can be though of as an extension of logical scaling based on description logic.

Relational scaling is based on a set of contexts $\{\langle G_x, M_x, I_x \rangle\}_{x \in X}$, the corresponding lattices $\{L_x\}_{x \in X} = \{FCA^*(\langle G_x, M_x, I_x \rangle)\}_{x \in X}$ and a set of relations $R = \{r_y\}_{y \in Y}$. This input, denoted by Σ, can be encoded as sets of description logic axioms by:

$$
\begin{aligned}
\lvert K_x \rvert &= \{m(g) \mid m \in M_x \wedge g \in G_x \wedge g I_x m\} \\
\lvert r_y \rvert &= \{r_y(g, g') \mid g \in G_x \wedge g' \in G_z \wedge \langle g, g' \rangle \in r_y\} \\
\lvert L_x \rvert &= \{c(g) \mid c \in L_x \wedge g \in extent(c)\} \\
\lVert L_x \rVert &= \{c \equiv \sqcap_{d \in intent(C)} d \mid c \in L_x\}
\end{aligned}
$$

The elements in $\lvert \cdot \rvert$ are part of an ABox and those in $\lVert \cdot \rVert$ are part of a TBox. They may be combined into a description logic knowledge base $\Sigma = \langle T_\Sigma, A_\Sigma \rangle$ such that:

$$
\begin{aligned}
T_\Sigma &= \bigcup_{x \in X} \lVert L_x \rVert \\
A_\Sigma &= \bigcup_{x \in X} \lvert K_x \rvert \cup \bigcup_{y \in Y} \lvert r_y \rvert \cup \bigcup_{x \in X} \lvert L_x \rvert
\end{aligned}
$$

The attributes provided by relational scaling are description logic concept descriptions, i.e. unary predicates. They can be interpreted with respect to the knowledge base associated to Σ:

$$\Sigma \vDash gI \; \forall \exists \; p.c \text{ iff } \Sigma \vDash (\forall p.c \sqcap \exists p)(g)$$

This way of interpreting relational scaling opens the door to introducing arbitrary description logic axioms within the scaling operation and thus to use background knowledge.

2.2.6 Other extensions

There are other extensions of formal context analysis for providing it with more expressiveness without scaling. We mention them here briefly.

Pattern structures

It is also possible to avoid scaling and to directly work on complex data, using the formalism of "pattern structures" [Ganter and Kuznetsov 2001; Kaytoue et al. 2011]. Pattern structures are a generalisation of FCA in which 2^M is replaced by elements of a meet-semilattice $\langle D, \sqcap \rangle$ [Ganter and Kuznetsov 2001; Kuznetsov 2009]. The formal context is now $\langle G, D, \delta \rangle$ with $\delta : G \mapsto D$ a mapping.

In this case, the two operators $\cdot^{\uparrow}$ and $\cdot^{\downarrow}$ define a Galois connection between $\langle 2^G, \subseteq \rangle$ and $\langle D, \sqsubseteq \rangle$ with $A \subseteq G$ and $d \in D$:

$$A^{\uparrow} = \sqcap_{g \in A} \delta(g)$$
$$d^{\downarrow} = \{g \in G \mid d \sqsubseteq \delta(g)\}$$

such that $c \sqsubseteq d \equiv c \sqcap d = c$.

This requires to define: (a) how to order its elements $\sqsubseteq$ and (b) how to test that an object satisfies an attribute expression gId for $d \in D$. This can be rewritten as for scaling:

$$\langle D, \sqsubseteq \rangle \vDash gId \text{ iff } d \sqsubseteq \delta(g)$$

Pattern structures [Ganter and Kuznetsov 2001; Kuznetsov 2009] provide a more structured attribute language without scaling. However, its use is not directly related to the problem of context dependencies considered here as the attributes do not refer to concepts.

Relational extensions

On the contrary, other approaches [Kötters 2013; Ferré and Cellier 2020] aim at extracting conceptual structures from n-ary relations without resorting to scaling. Their concepts have intents that can be thought of as conjunctive queries and extents as tuples of objects, i.e. answers to these queries. Hence, instead of being classes, i.e. monadic predicates, concepts correspond to general polyadic predicates. For that purpose, they rely on more expressive input, e.g. in Graph-FCA [Ferré and Cellier 2020] the incidence relation is a hypergraph between objects, and produce a more expressive representation. A comparison of RCA and Graph-FCA is provided in [Keip et al. 2020]. Graph-FCA adopts a different approach than RCA but should, in principle, suffer from the same problem as the one illustrated here. However, intents would need to refer to concepts so created, i.e. named subqueries. This remains to be studied.

2.3 A very short introduction to RCA

Here, we briefly introduce the principles of relational concept analysis, which will be used to extract link key candidates.

Relational Concept Analysis (RCA) [Rouane-Hacene et al. 2013a] extends FCA to the processing of relational datasets and allows inter-object relations to be materialised and incorporated into formal concept intents. RCA is a way to induce a description logic TBox from a simple ABox [Baader et al. 2003], using specific scaling operations. It may also be though of as a general way to deal with circular references using different scaling operations.

2.3.1 Operations

RCA applies to a relational context[2] $\langle K^0, R\rangle$, composed of a family of formal contexts $K^0 = \{\langle G_x, M_x^0, I_x^0\rangle\}_{x\in X}$ and a set of binary relations $R = \{r_y\}_{y\in Y}$. A relation $r_y \subseteq G_x \times G_z$ connects two object sets, a domain G_x ($dom(r_y) = G_x$, $x \in X$) and a range G_z ($ran(r_y) = G_z$, $z \in X$).

RCA applies relational scaling operations from a set Ω to each $K_x^i \in K^i$ and all relations $r_y \subseteq G_x \times G_z$ from the set of concepts in corresponding $L_z = FCA(K_z^i)$.

For performing its operations, RCA thus relies on FCA and σ_ς. More precisely it uses FCA^* and σ_Ω^* defined as:

$$FCA^*(\{\langle G_x, M_x, I_x\rangle\}_{x\in X}) = \{FCA(\langle G_x, M_x, I_x\rangle)\}_{x\in X}$$

$$\sigma_\Omega^*(\{\langle G_x, M_x, I_x\rangle\}_{x\in X}, R, \{L_x\}_{x\in X}) = \left\{ \bigoplus_{r_y\in R \mid r\subseteq G_x\times G_z}^{\varsigma\in\Omega} \sigma_\varsigma(\langle G_x, M_x, I_x\rangle, r_y, L_z) \right\}_{x\in X}$$

such that $\oplus_{r_y\in R \mid r\subseteq G_x\times G_z}^{\varsigma\in\Omega}$ scales, with all operations in Ω, the given context with all the relations starting from x (to any z).

2.3.2 Algorithm

RCA starts from the initial formal context family K^0 and thus iterates the application of the two operations:

$$K^{i+1} = \sigma_\Omega^*(K^i, R, FCA^*(K^i))$$

until reaching closure, i.e. reaching n such that $K^{n+1} = K^n$. Then, $RCA_\Omega(K^0, R) = FCA^*(K^n)$.

Thus, the RCA algorithm proceeds in the following way:

1. Initial formal contexts: $\{\langle G_x, M_x^0, I_x^0\rangle\}_{x\in X} \leftarrow \{\langle G_x, M_x, I_x\rangle\}_{x\in X}$.
2. $\{L_x^t\}_{x\in X} \leftarrow FCA^*(\{\langle G_x, M_x^t, I_x^t\rangle\}_{x\in X})$ (or, for each formal context, $\langle G_x, M_x^t, I_x^t\rangle$ the corresponding concept lattice $L_x^t = FCA(\langle G_x, M_x^t, I_x^t\rangle)$ is created using FCA).
3. $\{\langle G_x, M_x^{t+1}, I_x^{t+1}\rangle\}_{x\in X} \leftarrow \sigma_\Omega^*(\{\langle G_x, M_x^t, I_x^t\rangle\}_{x\in X}, R, \{L_x^t\}_{x\in X})$ (i.e. relational scaling is applied, for each relation r_y whose codomain lattice has new concepts, generating new contexts $\langle G_x, M_x^{t+1}, I_x^{t+1}\rangle$ including both plain and relational attributes in M_x^{t+1}).
4. If $\exists x \in X; M_x^{t+1} \neq M_x^t$ (scaling has occurred), go to Step 2.
5. Return: $\{L_x^t\}_{x\in X}$.

By abuse of notation, we note $\langle G, M, I\rangle \subseteq \langle G, M', I'\rangle$ whenever $M \subseteq M'$ and $I = I' \cap (G \times M)$. In this case, because I is the incidence relation between the same G and $M \subseteq M'$, the relation only depends on M and M'. This is generalised to formal context families $\{\langle G_x, M_x, I_x\rangle\}_{x\in X} \subseteq \{\langle G_x, M_x', I_x'\rangle\}_{x\in X}$ whenever $\forall x \in X, M_x \subseteq M_x'$.

2.3.3 Properties and semantics

The RCA process always reaches a closed formal context family for reason of finiteness [Rouane-Hacene et al. 2013a] and the sequence $(K^i)_{i=0}^n$ is non-(intent-)contracting, i.e. $\forall i \geq 0, K^i \subseteq K^{i+1}$ [Rouane-Hacene et al. 2013b].

The RCA semantics characterises the set of concepts in resulting RCA lattices as all and only those grounded on initial contexts (K_x^0) based on relations (R) [Rouane-Hacene et al. 2013b]. It thus can be considered as a well-grounded semantics: an attribute is scaled and applied to an object at iteration $i+1$ only if its condition applies at stage i. Hence, everything is ultmately relying on iteration K^0.

[2] We use the term 'relational context' instead of 'relational context family'.

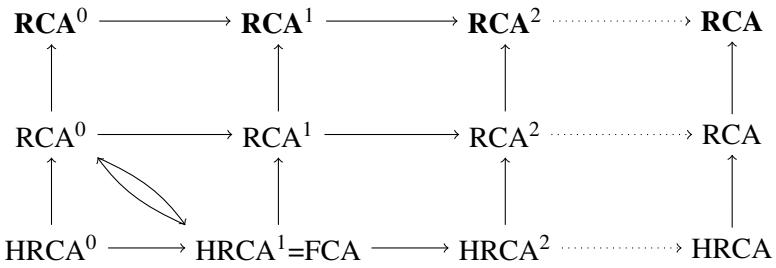


Figure 2.1: Relation between different restrictions of RCA (arrows mean: "can be rewritten into"). Bold labels denote the restriction with the same name interpreted with the semantics that will be developed here.

[Rouane-Hacene et al. 2013b] established that RCA indeed finds *the* K^n satisfying these constraints through correctness (the concepts of $FCA^*(K^n)$ are grounded in K^0 through R) and completess (all so-grounded concepts are in K^n).

2.4 RCA^0

In order to explain some specific phenomena in a clearer way, we will restrict some explanations to RCA^0, a special case of RCA. It is restricted in two ways:
- It contains only one formal context ($|X| = 1$),
- which has no attribute ($M_x^0 = \varnothing$).

Additionally, we will consider in the examples below only one single scaling operator: qualified existential scaling ($\Omega = \{\exists\}$).

In fact, it is possible to define several restrictions of RCA. They are noted RCA and FCA as usual, and HRCA for Hierarchical RCA when the dependency graph, whose edges are relations and vertices are contexts, is acyclic. In addition, an exponent indicates the maximum number of contexts authorised:
- FCA^0=HRCA^0: FCA with empty contexts is clearly not interesting
- FCA=HRCA^1: standard formal concept analysis
- HRCA: RCA with only hierarchical relations.
- RCA^0: RCA with one *empty* formal context
- RCA^1: RCA with one formal context
- RCA^n: RCA with n formal contexts
- RCA=$\exists n; RCA = RCA^n$

These are organised in Figure 2.1.

Because RCA^0 is a restriction of RCA, we will use the same notation as defined above, thought it operates on simpler structures.

In RCA^0, the process is simplified because $\Omega = \{\exists\}$ and there is only one formal context to deal with. It is thus:

$$L^0 = FCA(K^0)$$
$$L^{i+1} = FCA(\sigma^*_{\{\exists\}}(K^0, R, L^i))$$

Although RCA^0 seems very simple[3], FCA can be encoded into RCA^0. Indeed, given a formal context $\langle G, M, I\rangle$, for each attribute $m \in M$ in the formal context, a relation $R_m \subseteq G \times G$ can be created such that $\forall g \in G, \langle g, g\rangle \in R_m$ if and only if gIm. Starting with $K^0 = \langle G, \varnothing, \varnothing\rangle$, it can be checked that

[3] An anonymous ICFCA reviewer complements the remarks of §2.2.6 noting that RCA^0 is also very related to Graph-FCA as they both have only one context and using existential scaling.

$\sigma^*_\exists(K^0, R, FCA^*(K^0))$ will simply add to K^0 one attribute $\exists r_m.\top$ per $m \in M$ which will exactly correspond to m.

It is also possible to encode RCA^0 into FCA using the following trick: Given an RCA^0 relational context $\langle\{\langle G, \varnothing, \varnothing\rangle\}, \{R_p \subseteq G \times G\}_{p \in P}\rangle$, it can be encoded in a single FCA:

- G remains the same;
- $M = \{po | p \in P \wedge o \in G\}$;
- $o' I po$ iff $\langle o', o\rangle \in R_p$.

It is clear that all the information from the relational context has been preserved and FCA should be able to return an result analogous to RCA$^0_{\{\exists\}}$.

Introducing RCA^0 is sufficient to hint at the problems and solutions that we want to illustrate.

Conclusion

We introduced the necessary material to deal with cyclic dependencies in link key extraction using FCA extensions. We will now consider how this can be achieved (Chapter 3) and what issues this raises (Chapter 4).

3. Dealing with cyclic dependent link keys through relational concept analysis

In Deliverable 1.1, we provided algorithms able to extract hierarchically dependent link keys [Euzenat et al. 2019]. However, classes are not always hierarchically organised. Object properties may loop through the set of classes. Even if this does not affect resulting link key candidates, this cannot be ruled out.

We show how to extend the definitions of Deliverable 1.1 to account for cyclic dependencies and extracting dependent link keys.

3.1 RCA encoding of the extraction of dependent link key candidate

To solve this problem we used RCA [Rouane-Hacene et al. 2013a]. Accordingly, the problem is modelled, as in RCA, through the set of formal contexts corresponding to pairs of classes and a set of relational contexts corresponding to pairs of object properties. We thus diverge from the definition of formal contexts for independent link key candidates, provided in Definition 11 of Deliverable 1.1, by initially recording only datatype properties within the formal contexts (as in Definition 9 of Deliverable 1.1) and using relational contexts for object properties.

Definition 1 (Relational contexts for dependent link key candidates)**.** *Given two datasets D of signature $\langle R,P,C\rangle$ and D' of signature $\langle R',P',C'\rangle$ and an alignment $A \subseteq C \times C'$, given two pairs of classes $\langle c,c'\rangle$ and $\langle d,d'\rangle$ of A and a pair of object properties $\langle r,r'\rangle$ of $R \times R'$, the relational context associated with r and r' in D and D' is defined by $\langle c^D \times c'^{D'}, d^D \times d'^{D'}, I_{\langle r,r'\rangle}\rangle$ such that:*

$$\langle x,x'\rangle \; I_{\langle r,r'\rangle} \; \langle y,y'\rangle \;\; \textit{iff} \;\; y \in r^D(x) \;\; \textit{and} \;\; y' \in r'^{D'}(x')$$

The main difference with RCA is that RCA is designed for extracting conceptual descriptions of classes, though we aim at extracting link key candidates. The former describes classes, the latter describes how to identify instances. Another difference is that coherent families of link key candidates have no analogous notion in RCA as all class descriptions may coexist: no compatibility is requested. However, at the technical level of concept discovery, the process is the same. In particular, the relational contexts are exactly the same. They are simply extended to pairs of properties and thus pairs of instances.

Scaling operators for generating link key conditions were defined in [Atencia et al. 2020] as follows.

Definition 2 (Link key condition scaling operators)**.** *Given two datasets D of signature $\langle R,P,C\rangle$ and D' of signature $\langle R',P',C'\rangle$, and a relational context $\langle c^D \times c'^{D'}, d^D \times d'^{D'}, I_{\langle r,r'\rangle}\rangle$ for a pair of object properties $\langle r,r'\rangle \in R \times R'$,*

- *the universal scaling operator $f^{\forall}_{r,r'} : (c^D \times c'^{D'}) \times \mathcal{K}_{r,r'} \to \mathbb{B}$ is defined by $f^{\forall}_{r,r'}(\langle o,o'\rangle,k)$ iff $r^D(o) =_k r'^{D'}(o')$,*
- *the existential scaling operator $f^{\exists}_{r,r'} : (c^D \times c'^{D'}) \times \mathcal{K}_{r,r'} \to \mathbb{B}$ is defined by $f^{\exists}_{r,r'}(\langle o,o'\rangle,k)$ iff $r^D(o) \cap_k r'^{D'}(o') \neq \varnothing$.*

We can now redefine them in the framework of Section 2.2.1.

Definition 3 (Link key condition scaling operators)**.** *Given two datasets D of signature $\langle R,P,C\rangle$ and D' of signature $\langle R',P',C'\rangle$ and $L_{C\times C'}$ a structure associating a concept lattice with each pair of classes in $C \times C'$, let $\langle r,r'\rangle \in R \times R'$ be a pair of relations, $\langle o,o'\rangle \in \mathrm{dom}(r) \times \mathrm{dom}(r')$ a pair of objects and k a concept of the lattice associated to $\mathrm{ran}(r) \times \mathrm{ran}(r')$ by $L_{C\times C'}$,*

$$\langle R \times R', L_{C\times C'}\rangle \vDash \langle o,o'\rangle I \forall \langle r,r'\rangle_k \textit{ iff } r^D(o) =_k r'^{D'}(o')$$
$$\langle R \times R', L_{C\times C'}\rangle \vDash \langle o,o'\rangle I \exists \langle r,r'\rangle_k \textit{ iff } r^D(o) \cap_k r'^{D'}(o') \neq \varnothing$$

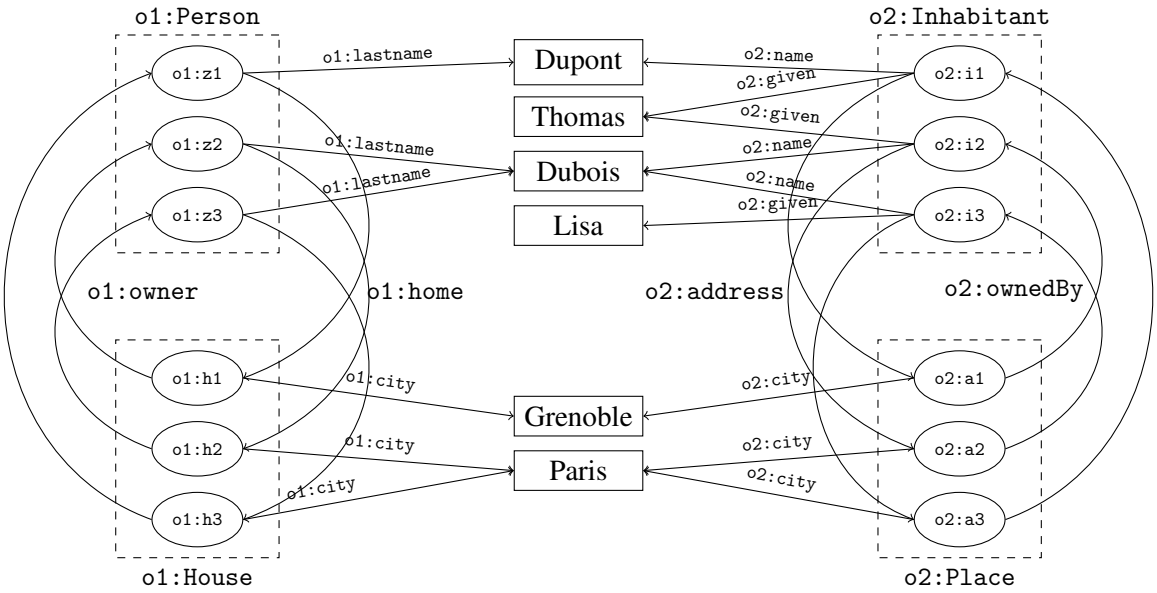


Figure 3.1: Datasets illustrating mutual dependencies between classes.

The universal and existential scaling operators scale the formal contexts of Definition 9 of Deliverable 1.1 upon the $\forall$-conditions and $\exists$-conditions on the object property pairs of the relational contexts in Definition 1.

Hence, a relational context [Rouane-Hacene et al. 2013a] can be associated to two datasets and an alignment:

Definition 4 (Relational context for dependent link key candidates)**.** *Given two datasets D of signature $\langle R,P,C\rangle$ and D' of signature $\langle R',P',C'\rangle$, and an alignment $A \subseteq C \times C'$, the relational context for dependent link key candidate extraction between D and D' through A is composed of:*

- *all the formal contexts corresponding to pairs of classes in A,*
- *all the relational contexts corresponding to object property pairs in $R \times R'$ relating pairs of classes from A.*

The solution to the problem is a collection of concept lattices, one per pair of aligned classes in A, from which the coherent families of link key candidates may be extracted exactly like in Section 6 of Deliverable 1.1. The algorithm is that of relational concept analysis, using the link key condition scaling operators (Definition 2 or 2):

1. Start with the formal contexts corresponding to aligned classes.
2. Apply formal concept analysis to all contexts.
3. Apply the scaling operators on the relational contexts to add columns in the formal contexts depending on the already extracted keys.
4. If scaling has occurred, go to Step 2.
5. Extract the coherent families by picking one link key candidate per pair of classes in A, as long as they are compatible.

The link set associated with a coherent family of link keys is the set of links in the extent of these link keys.

Example 1 illustrates the extraction of cyclic link key candidates.

Example 1 (Cyclic link key candidate extraction)**.** *Figure 3.1 provides an example of two datasets containing circular dependencies. In the first, left-hand side, dataset, classes* `Person` *and* `House` *mutually depend on each other through relations* `home` *and* `owner`*. Respectively, in the second, right-hand side,*

	$\exists\langle lastname, given\rangle$	$\exists\langle lastname, name\rangle$	$\forall\langle lastname, given\rangle$	$\forall\langle lastname, name\rangle$
$\langle z_1, i_1\rangle$		×		×
$\langle z_1, i_2\rangle$				
$\langle z_1, i_3\rangle$				
$\langle z_2, i_1\rangle$				
$\langle z_2, i_2\rangle$		×		×
$\langle z_2, i_3\rangle$		×		×
$\langle z_3, i_1\rangle$				
$\langle z_3, i_2\rangle$		×		×
$\langle z_3, i_3\rangle$		×		×

	$\exists\langle city, city\rangle$	$\forall\langle city, city\rangle$
$\langle h_1, a_1\rangle$	×	×
$\langle h_1, a_2\rangle$		
$\langle h_1, a_3\rangle$		
$\langle h_2, a_1\rangle$		
$\langle h_2, a_2\rangle$	×	×
$\langle h_2, a_3\rangle$	×	×
$\langle h_3, a_1\rangle$		
$\langle h_3, a_2\rangle$	×	×
$\langle h_3, a_3\rangle$	×	×

Figure 3.2: Formal contexts for the pairs of classes $\langle$`Person`,`Inhabitant`$\rangle$ and $\langle$`House`,`Place`$\rangle$ in Figure 3.1.

$R_{owner,ownedBy}$	$\langle z_3, i_1\rangle$	$\langle z_3, i_2\rangle$	$\langle z_3, i_3\rangle$	$\langle z_2, i_1\rangle$	$\langle z_2, i_2\rangle$	$\langle z_2, i_3\rangle$	$\langle z_1, i_1\rangle$	$\langle z_1, i_2\rangle$	$\langle z_1, i_3\rangle$
$\langle h_1, a_1\rangle$					×				
$\langle h_1, a_2\rangle$						×			
$\langle h_1, a_3\rangle$				×					
$\langle h_2, a_1\rangle$		×							
$\langle h_2, a_2\rangle$			×						
$\langle h_2, a_3\rangle$	×								
$\langle h_3, a_1\rangle$								×	
$\langle h_3, a_2\rangle$									×
$\langle h_3, a_3\rangle$							×		

$R_{home,address}$	$\langle h_3, a_2\rangle$	$\langle h_3, a_3\rangle$	$\langle h_3, a_1\rangle$	$\langle h_2, a_2\rangle$	$\langle h_2, a_3\rangle$	$\langle h_2, a_1\rangle$	$\langle h_1, a_2\rangle$	$\langle h_1, a_3\rangle$	$\langle h_1, a_1\rangle$
$\langle z_1, i_1\rangle$									×
$\langle z_1, i_2\rangle$							×		
$\langle z_1, i_3\rangle$								×	
$\langle z_2, i_1\rangle$						×			
$\langle z_2, i_2\rangle$				×					
$\langle z_2, i_3\rangle$					×				
$\langle z_3, i_1\rangle$			×						
$\langle z_3, i_2\rangle$	×								
$\langle z_3, i_3\rangle$		×							

Figure 3.3: Relational contexts for pairs of relations $\langle$`owner`,`ownedBy`$\rangle$ and $\langle$`home`,`address`$\rangle$ in Figure 3.1.

dataset, classes `Inhabitant` *and* `Place` *also mutually depend on each other through relations* `address` *and* `ownedBy`*. The relational context is made of formal contexts presented in Figure 3.2 and relations given in Figure 3.3. It is iteratively scaled using the scaling operators of Definition 3. It converges to a fixed point after six iterations. The final lattices are given in Figure 3.4. We obtain two pairs of compatible concepts, namely* $C1-C2$ *and* $C4-C6$*. The coherent family of link key candidates* $C4-C6$ *generates all and only relevant links.*

3.2 Implementation and complexity considerations

The prototype described in Deliverable 1.1[1] is able to deal with relational concept analysis.

The RCA processing has been implemented by iteratively applying the scaling operators to the formal contexts.

[1] The implementation is available from https://moex.inria.fr/software/linkky/.

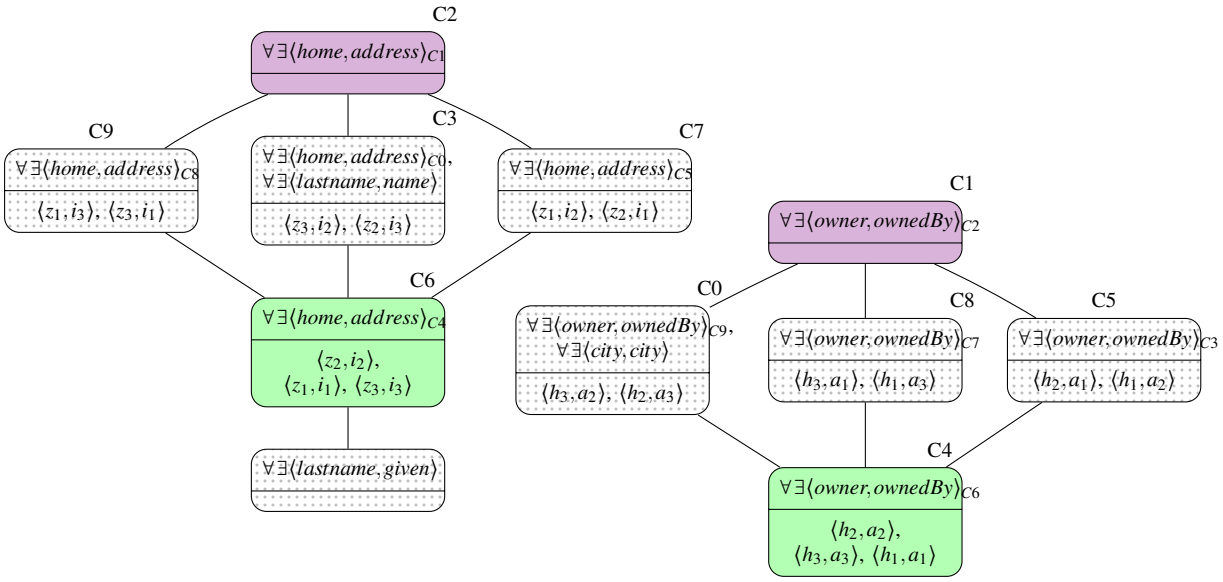


Figure 3.4: Concept lattices obtained from the relational context of Figures 3.2 and 3.3 after six iterations.

The example has been computed by the developed system. The relational contexts and concept lattices have been directly generated by this implementation (we only changed node colours and patterns for legibility).

The remarks on complexity and scalability apply to this work as well. It is very likely that the worst-case complexity of our algorithms is that of RCA.

Using RCA, the number of relational attributes may be higher than $4^{n_P^2}$ since it depends on the number of concepts in the range lattices. It remains however bounded by $2^{n_I^2}$. Hence, the worst-case complexity should still be exponential in $O(n_I^4 2^{n_I^2})$ (n_I is the number of individuals in one dataset and n_P is the number of properties and relations).

Conclusion

The use of relational concept analysis provides a satisfactory answer to the extraction of circularly dependent link keys. However, the link key lattices are relatively reduced and it seems that more risks could be taken (and evaluated later by measures). This led us to reconsider the semantics of RCA in order to offer more link key candidates.

4. Problems with circular dependencies

Problems may occur when using RCA on data sets more dependent on relations than attributes. These are illustrated here based on RCA^0. Moreover, all examples are provided with one single scaling operator (which may differ from example to example).

The essence of the considered examples is as follows: Consider a population of individuals about which nothing is known except that some are married to others, and some have others as child. Can we classify them as parents, as spouses, as married parents? We can, but apparently RCA cannot unless it knows about their favorite pizza topping.

4.1 Base example

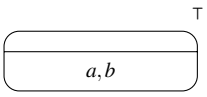
Consider a single class: Person with attribute spouse (sp)

$$A = \{\top(a), \top(b), sp(a,b), sp(b,a)\}$$

Can be encoded as an empty formal context for $\top$ and the relational self-context:

R_{sp}	a	b
a		×
b	×	

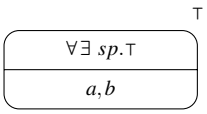
The empty context will generate the single lattice:



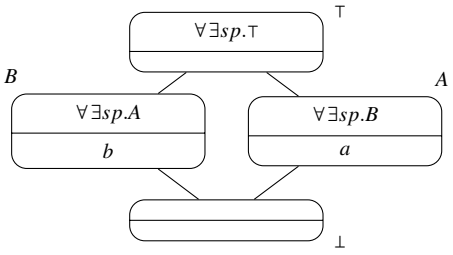
The application of the scaling operation provides the attribute $\forall\exists\, sp.\top$ which generates a new context:

	$\forall\exists\, sp.\top$
a	×
b	×

Leading to the final lattice:



However, the following lattice would not be less good:



This corresponds to two different knowledge bases:

$$T_1 = \{A \sqsubseteq \forall\exists\, sp.B, B \sqsubseteq \forall\exists\, sp.A\}$$
$$A_1 = \{A(a), B(b), sp(a,b), sp(b,a)\}$$

and

$$T_2 = \{\top \sqsubseteq \forall\exists\, sp.\top\}$$
$$A_2 = \{\top(a), \top(b), sp(a,b), sp(b,a)\}$$

The former one, may not be the most stupid one, especially if one considers that A is *Woman* and B is *Man* for instance (yes, a very conservative interpretation).

Hence the question: if the goal of RCA is to extract class descriptions, why does it only generate the second one? The answer may be semantic or technical. We assume that semantically, it extracts least fixed-point. Technically, it uses an intent-agglomerative/extent-divisive algorithm which will agglomerate concept description and, thus, separate their instances. Since it starts with a set containing all objects, if there is no reason to separate two objects, they will remain in the same extent.

4.2 Running example

Consider the following ABox:

$A = \{\top(a), \top(b), \top(c), \top(d), p(a,b), p(b,a), p(c,d), p(d,c), p(a,a), p(b,b)\}$

This can be encoded as an empty formal context and the relation of Figure 4.1 (left). The empty context will generate the single lattice of Figure 4.1 (right) (names are assigned to concepts according to their extent).

Scaling with $\exists$ and p provides the attribute $\exists p.ABCD$ which generates the new context of Figure 4.2 (left), leading to the lattice of Figure 4.2 (right) which is the one returned by RCA.

However, the concept lattices of Figure 4.3 are other valid lattices worth considering.

They correspond to different knowledge bases:

$$T_1 = \{ABCD \sqsubseteq \exists p.ABCD\}$$
$$A_1 = \{ABCD(a), ABCD(b), ABCD(c), ABCD(d),$$
$$p(a,b), p(b,a), p(c,d), p(d,c), p(a,a), p(b,b)\}$$

and

$$T_2 = \{AB \sqsubseteq \top \sqcap \exists p.AB, CD \sqsubseteq \top \sqcap \exists p.CD, ABCD \sqsubseteq \exists p.ABCD\}$$
$$A_2 = \{AB(a), AB(b), CD(c), CD(d), p(a,b), p(b,a), p(c,d), p(d,c), p(a,a), p(b,b)\}$$

p	a	b	c	d
a	×	×		
b	×	×		
c				×
d			×	

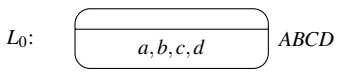


Figure 4.1: Relation (left) and initial concept lattice (right).

	$\exists p.ABCD$
a	×
c	×
b	×
d	×

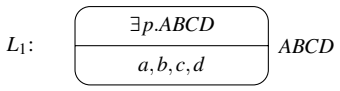


Figure 4.2: Scaled context (left) and final concept lattice L_1 (right).

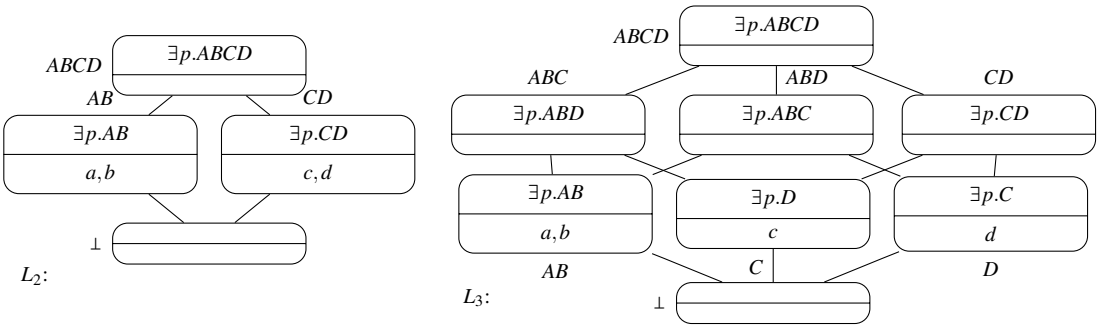


Figure 4.3: Alternative concept lattices (L_2 and L_3).

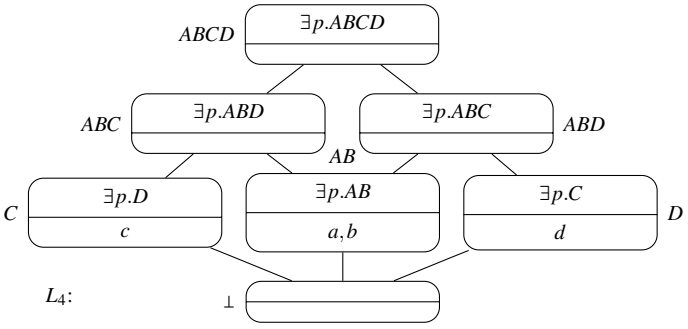


Figure 4.4: Another alternative concept lattices (L_4).

and

$$T_3 = \{AB \sqsubseteq ABC \sqcap ABD \sqcap \exists p.AB, C \sqsubseteq ABC \sqcap CD \sqcap \exists p.D, D \sqsubseteq ABD \sqcap CD \sqcap \exists p.C,$$
$$ABC \sqsubseteq ABCD \sqcap \exists p.ABD, ABD \sqsubseteq ABCD \sqcap \exists p.ABC, CD \sqsubseteq ABCD \sqcap \exists p.CD,$$
$$ABCD \sqsubseteq \exists p.ABCD\}$$
$$A_3 = \{AB(a), AB(b), C(c), D(d), p(a,b), p(b,a), p(c,d), p(d,c), p(a,a), p(b,b)\}$$

In addition to extracting the TBox, these extend the ABox. However, in RCA and FCA, objects are also assigned to the created concepts. In this case, this assignment has consequences on the scaled attributes taken into account and hence the resulting lattice.

As in classical RCA, each concept of these lattices is closed with respect to the specific formal context scaled by $\exists$ and p from the concepts of the lattice. Moreover, the lattices are self-supported in the sense that their attributes refer only to the concepts of the lattice.

4.3 Interleaved references

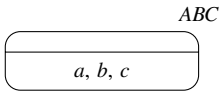
Here is an example showing the effect of longer dependencies (it is written with $\forall\exists$ but would work with $\exists$). Consider a single class $\top$ with attributes p. The ABox is provided as:

$$ABox = \{\top(a), \top(b), \top(c), p(a,b), p(b,c), p(c,a)\}$$

Which will be encoded in the empty formal context for $\top$ and the relational self-context:

R_p	a	b	c
a			×
b	×		
c		×	

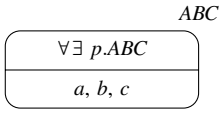
The empty context will generate the single lattice:



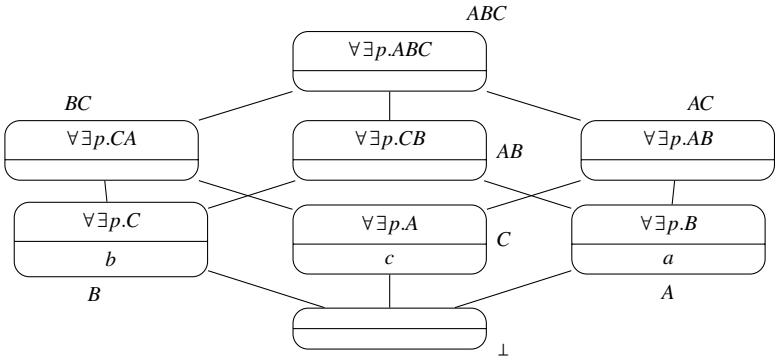
The application of the scaling operation provides the attribute $\forall\exists\, p.\top$ which generates a new context:

	$\forall\exists\, p.ABC$
a	×
b	×
c	×

Leading to the final lattice:



Here, all the concepts of the powerset lattice are closed.



4.4 Link key extraction example

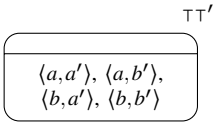
Consider a single pair-of-classes $\top\top'$ with attributes p and p' (let say spouse and conjoint). Our ABox is provided as:

$$A = \{\top(a), \top(b), p(a,b), p(b,a), \top'(a'), \top'(b'), p'(a',b'), p'(b',a')\}$$

Which will be encoded in the empty formal context for $\langle \top, \top' \rangle$ and the relational self-context:

$R_{\langle p,p' \rangle}$	$\langle a,a' \rangle$	$\langle a,b' \rangle$	$\langle b,a' \rangle$	$\langle b,b' \rangle$
$\langle a,a' \rangle$				×
$\langle a,b' \rangle$			×	
$\langle b,a' \rangle$		×		
$\langle b,b' \rangle$	×			

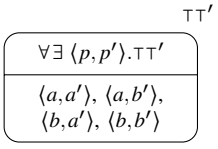
The empty context will generate the single lattice:



The application of the scaling operation provides the attribute $\forall\exists\, \langle p,p' \rangle$ which generates a new context:

	$\forall\exists\, \langle p,p' \rangle.\top\top'$
$\langle a,a' \rangle$	×
$\langle a,b' \rangle$	×
$\langle b,a' \rangle$	×
$\langle b,b' \rangle$	×

Leading to the final lattice:



concept	discriminability	F-measure	coverage
$ABCD$	.5	.66	1
AB	1	1	1
CD	1	1	1
B	1	.66	.5
A	1	.66	.5
D	1	.66	.5
C	1	.66	.5
$\bot$	1	0	0

Table 4.1: Discriminability, F-measure and coverage of all concepts of the lattice of Section 4.4.

From the coverage standpoint this is quite nice because we obtain links for every objects. However, this is quite bad from the discriminability standpoint: all objects are equal!

Here again, the link keys identified in the following graph may be more interesting:

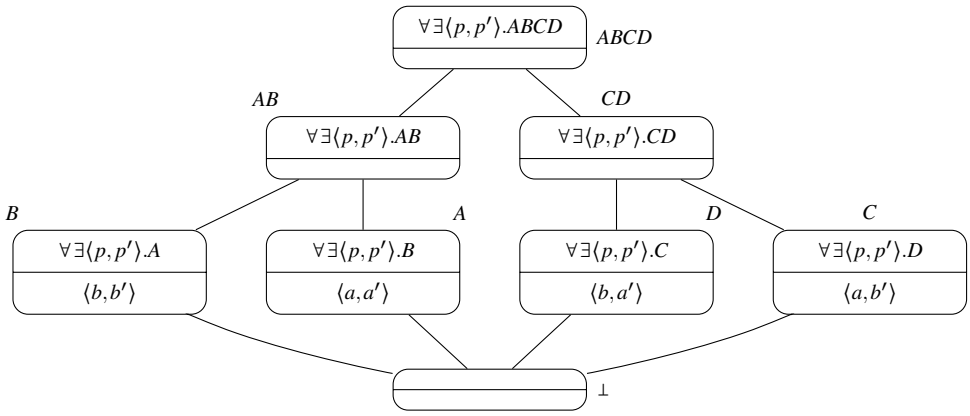


Table 4.1 shows the discriminability and coverage measures [Atencia et al. 2014] of all concepts of this lattice. It shows that $ABCD$, the only concept returned by RCA, is not necessarily the best link key (f-measure=.66), and that concepts AB and CD have a perfect discriminability and coverage.

The reader can observe that the example provided for concept extraction equally applies for link key extraction. It is sufficient to find which pairs of data sets this encodes.

Conclusion

As shown in Chapter 3, it is possible to extract link keys. However, the examples above show that not all link key candidates are found. This is a problem because the algorithm is supposed to judge on them all which one is the best (Deliverable 1.3, [Atencia et al. 2014]).

We have restricted ourselves to simple examples, however it should even be possible to find examples in which RCA may be used to extract both description logic classes and link keys. In such a case, link key candidates may depend on induced classes and induced classes may depend on link key candidates.

The problem applies to RCA as a whole as RCA^0 is included in RCA. Hence the question: Why does RCA returns only one lattice, and which one? Answering it requires to reconsider the RCA semantics.

Hence, we will try to address the following questions:

– Which of those solution are extracted by RCA algorithms? (Chapter 5)
– Is it possible to obtain them all? (Chapter 6)

For this, it is necessary to dig deeper in RCA semantics.

5. A fixed-point semantics for RCA^0

In order to investigate the semantics of relational concept analysis, we adopt a functional standpoint in which RCA is defined as a function in a precisely defined space. In Sections 5.1 and 5.2 we provide two alternative, and equivalent, characterisations of that space, which provide the semantics for RCA^0. In Section 5.3.1, we relate the well-grounded RCA semantics to these two semantics by showing that RCA computes the least fixed point of these functions. We then discuss a further extension of that semantics intended to better characterise the interesting fixed points that could be considered by an extension of RCA. This is achieved through the notion of self-supported lattices (Section 5.4) which is characterised as the fixed points of another function that can be seen as complementary to that used by RCA. Finally, we discuss the precise characterisation of the space of interesting solutions, fixed points of these complementary functions, by considering the composition of the corresponding closures (Section 5.5).

Warning: a nest of fixed points FCA is a domain of fixed points, hence it is easy to get lost among the various fixed points involved:
- In description logics, on which RCA relies, the semantics of concepts is given by (least) fixed points when circularities occur [Nebel 1990];
- FCA's goal is to compute fixed points: concepts are the result of a closure operator which is also a fixed point [Belohlávek 2008];
- finally, when confronted to cycles, the RCA concept lattice is the fixed point of the function that grows a lattice family from the previous one.

The present work is concerned with the fixed points of the latter function.

5.1 Semantics and properties: the context approach

We first adopt the standpoint of the formal contexts. We define precisely the space of contexts in which RCA is computed and the functions underlying RCA in that space.

5.1.1 The lattice $\mathcal{K}$ of RCA^0 contexts

The contexts considered by RCA are formal context families scaled from the initial context using the scaling operations. They are determined by three elements given once and for all: $K^0 = \{\langle G_x, M_x^0, I_x^0\rangle\}_{x\in X}$, $R = \{r_y\}_{y\in Y}$, and Ω. This is even more specific for RCA^0 with $K^0 = \langle G, \varnothing, \varnothing\rangle$ and $\Omega = \{\exists\}$, but for most of this section we will ignore it.

Through these operations, only M_x^i and I_x^i change, the latter depending directly from the former (Property 1).

Property 1 (The incidence matrix depends only on the relations)**.** *For a relational scaling operation ς and a relation $r \subseteq G_x \times G_z$, an attribute $m \in D_{\varsigma,r,N(G_z)}$ determines the incidence on objects of G_x.*

Proof. m is scaled from a scaling operator ς, a relation r and a concept C (eventually a cardinal n). From Table 2.3, it only depends on ς, r and the extent of C. However, ς and r are the same in all situations, they are not interpreted contextually. The concept C is identified by a name which denotes its extent. Hence, its extent is not depending on the context either. So wether an object of G_x satisfies this attribute or not depends solely on the attribute. □

The attribute language $D_{\Omega,R,N(G)}$ is fully determined by the non-changing parts G_x, R and Ω. More precisely, a finite set of relations R and scaling operations Ω determines the finite set $D_{\Omega,R,N(G)} = \bigcup_{r\in R}^{\varsigma\in\Omega} D_{\varsigma,r,N(G)}$ of possible scaled attributes in RCA^0.

Hence, the formal contexts considered by RCA are those obtained by adding subsets of $D_{\Omega,R,N(G)}$:

$$\mathcal{K}_{\langle G,M^0,I^0\rangle,R,\Omega} = \{K_{+M}^{\langle R,N(G)\rangle}(\langle G,M^0,I^0\rangle) \mid M \subseteq D_{\Omega,R,N(G)}\}$$

with $K_{+M}^{\langle R,N(G)\rangle}(.)$ the operation defined in §2.2.1.

Given $K,K' \in \mathcal{K}_{\langle G,M^0,I^0\rangle,R,\Omega}$ such that $K = \langle G,M^0 \cup M, I^0 \cup I\rangle$ and $K' = \langle G,M^0 \cup M', I^0 \cup I'\rangle$, $K \vee K'$ and $K \wedge K'$ are defined as:

(join) $$K \vee K' = \langle G,M^0 \cup (M \cup M'), I^0 \cup (I \cup I')\rangle$$

(meet) $$K \wedge K' = \langle G,M^0 \cup (M \cap M'), I^0 \cup (I \cap I')\rangle$$

$\mathcal{K}_{K^0,R,\Omega}$ is thus closed by meet and join.

Property 2. *$\langle \mathcal{K}_{K^0,R,\Omega}, \vee, \wedge\rangle$ is a complete lattice.*

Proof. $\vee$ and $\wedge$ satisfy commutativity, associativity and the absorption laws directly from the union and intersection on sets, so this is a lattice. It is complete because finite. □

Property 3. *$\forall K,K' \in \mathcal{K}_{K^0,R,\Omega}, K \subseteq K'$ iff $K = K \wedge K'$*

Proof. This property also comes directly from its set theoretic counterpart application to M and M': $K \subseteq K' \Leftrightarrow M \subseteq M' \Leftrightarrow M = M \cap M' \Leftrightarrow K = K \wedge K'$ □

5.1.2 The context expansion function F

We reformulate RCA as based on a main single function, $F_{K^0,R,\Omega}$, the context expansion function[1] attached to a relational context $\langle K^0,R\rangle$ and a set Ω of scaling operations.

Definition 5 (Context expansion function)**.** *Given a relational context $\langle K^0,R\rangle$ and a set of relational scaling operations Ω, the function $F_{K^0,R,\Omega} : \mathcal{K}_{K^0,R,\Omega} \mapsto \mathcal{K}_{K^0,R,\Omega}$ is defined by:*

$$F_{K^0,R,\Omega}(K) = \sigma_\Omega^*(K,R,FCA^*(K)))$$

The function expression is independent from K^0, K^0 is used to restrict the domain of the function so that its elements cover K^0. From now on, we will abbreviate $\mathcal{K}_{K^0,R,\Omega}$ as $\mathcal{K}$ and $F_{K^0,R,\Omega}$ as F. This is legitimate because, for a given relational context, K^0, R and Ω do not change. F is an extensive and monotone internal operation for $\mathcal{K}$:

Property 4. *$\forall K \in \mathcal{K}$, $F(K) \in \mathcal{K}$*

Proof. Scaling only adds attributes from $D_{\Omega,R,N(G)}$. □

Property 5 (F is extensive and monotone)**.** *The function F attached to a relational context and a set of scaling operator satisfies:*

(extensivity) $$K \subseteq F(K)$$

(monotony) $$K \subseteq K' \Rightarrow F(K) \subseteq F(K')$$

Proof. extensivity holds because F eventually adds to each formal context in K new attributes scaled from $FCA^*(K)$. The set of attributes can thus not be smaller. monotony holds because $K \subseteq K'$ means that $M \subseteq M'$. This entails that the set of concepts of $FCA^*(K)$ is included in that of $FCA^*(K')$, hence the set of attributes A scaled from K is included in the set A' scaled from K'. Since, they are added to M and M', then $M \cup A \subseteq M' \cup A'$, hence $F(K) \subseteq F(K')$. □

Extensivity corresponds to the non-contracting property of the well-grounded semantics [Rouane-Hacene et al. 2013b] and monotony is also called order-preservation.

[1] Named "complete relational extension" in [Rouane-Hacene et al. 2013b].

5.1.3 Fixed points of F

Given F, it is possible to define its sets of fixed points, i.e. the sets of formal contexts closed for F, as:

Definition 6 (fixed point)**.** *A formal context $K \in \mathcal{K}$ is a fixed point for a context expansion function F, if $F(K) = K$. We call $fp(F)$ the set of fixed points for F.*

We can define:

$$lfp(F) = \bigwedge_{K \in fp(F)} K \text{ and } gfp(F) = \bigvee_{K \in fp(F)} L$$

Since $\mathcal{K}$ is a complete lattice and F is order-preserving (or monotone) on $\mathcal{K}$, then the Knaster-Tarski theorem applies:

Theorem 6 (Knaster-Tarski theorem [Tarski 1955])**.** *Let $\mathcal{K}$ be a complete lattice and let $F : \mathcal{K} \mapsto \mathcal{K}$ be an order-preserving function. Then the set of fixed points of F in $\mathcal{K}$ is also a complete lattice.*

This theorem applies to *any* complete lattice and order-preserving function. We used $\mathcal{K}$ and F because this is to what it is applied here. It will be latter applied to

In particular, this warrants that there exists least and greatest fixed points of F in $\mathcal{K}$ (called $lfp(F)$ and $gfp(F)$).

5.2 Semantics and properties: the lattice approach

In this section, we approach RCA from the lattice standpoint and we show, unsurprisingly, that it is remarkably parallel to the context approach.

5.2.1 The lattice $\mathcal{L}$ of RCA^0 concept lattices

From $\mathcal{K}_{K^0,R,\Omega}$, one can define $\mathcal{L}_{K^0,R,\Omega}$ as the finite set of images of $\mathcal{K}_{K^0,R,\Omega}$ by FCA*. These are concept lattices obtained by applying FCA* on K^0 extended with a subset of $D_{\Omega,R,N(G)}$:

$$\mathcal{L}_{\langle G,M^0,I^0\rangle,R,\Omega} = \{FCA^*(\langle G, M^0 \cup M, I^0 \cup I\rangle) \mid M \subseteq D_{\Omega,R,N(G)}\}$$

We define a specific type of homomorphisms between two concept lattices when concepts are simply mapped into concepts with the same extent and possibly increased intent.

Definition 7 (Lattice homomorphism)**.** *A concept lattice homomorphism $h : \langle C, \leq\rangle \mapsto \langle C', \leq'\rangle$ is a function which maps each concept $c \in C$ of into a corresponding concept $h(c)$ such that:*

- *$\forall c \in C$, $intent(c) \subseteq intent(h(c))$ (or $intent(c) \supseteq intent(h(c))$ if these are considered as description logic concept descriptions),*
- *$\forall c \in C$, $extent(c) = extent(h(c))$, and*
- *$\forall c, d \in C$, $c \leq d \Rightarrow h(c) \leq' h(c)$.*

We note $L \preceq L'$ if there exists a homomorphism from L to L'. In principle, $L \simeq L'$ if $L \preceq L'$ and $L' \preceq L$, but here, $\simeq$ is $=$.

The order between concept lattices is straigthforwardly extended to families of concept lattices such that: $\{L_x\}_{x\in X} \preceq \{L'_x\}_{x\in X}$ iff $\forall x \in X$, $L_x \preceq L'_x$.

There exists an implicit function $\kappa : \mathcal{L}_{K^0,R,\Omega} \mapsto \mathcal{K}_{K^0,R,\Omega}$ such that $\forall L \in \mathcal{L}_{K^0,R,\Omega}$, $L = FCA(\kappa(L))$. Since $\simeq$ is the same as $=$ which identifies lattices containing concept having exactly the same intent and extent. $\kappa(L)$ can be induced by collecting the attributes present in L intents to build the unique M, from which the corresponding I is obtained [Ganter and Wille 1999]. It is straightforwardly generalised as

$$\kappa^*(\{L_x\}_{x\in X}) = \{\kappa(L_x)\}_{x\in X}$$

Because $K \subseteq K' \Rightarrow FCA^*(K) \preceq FCA^*(K')$, we can define $\wedge$ and $\vee$ on $\mathcal{L}_{K^0,R,\Omega}$. Given $L, L' \in \mathcal{L}_{\langle G,M^0,I^0\rangle,R,\Omega}$, such that $L = FCA^*(K)$ and $L' = FCA^*(K')$:

(join) $$L \vee L' = FCA^*(K \vee K')$$

(meet) $$L \wedge L' = FCA^*(K \wedge K')$$

It is clear that $\mathcal{L}_{K^0,R,\Omega}$ is closed by meet and join.

Property 7. *$\langle \mathcal{L}_{K^0,R,\Omega}, \vee, \wedge \rangle$ is a complete lattice.*

Proof. $\vee$ and $\wedge$ satisfy commutativity, associativity and the absorption laws directly from the union and intersection on (attribute) sets, so this is a lattice. It is complete because finite. □

Property 8. *$\forall L, L' \in \mathcal{L}_{K^0,R,\Omega}, L \preceq L'$ iff $L = L \wedge L'$*

Proof. First, for any $m \in D_{\Omega,R,N(G)}$ belonging to both M and M', the pairs of the incidence matrices I and I' for m are the same, because they are extracted from the same relations R (Property 1).

$\Rightarrow$ $L \preceq L'$ means that $\forall c \in L$, $\exists h(c) \in L'$ such that $extent(c) = extent(h(c))$ and $intent(c) \subseteq intent(h(c))$. This means that $M = \bigcup_{c \in L} intent(c) \setminus M^0$ and $M' = \bigcup_{c \in L'} intent(c) \setminus M^0$, hence $M \subseteq M'$ (and $I \subseteq I'$ due to Property 1). Hence, $L = L \wedge L'$ because the contexts on which they are built are the same.

$\Leftarrow$ $L = L \wedge L'$ means that $M \subseteq M'$ (and then $I \subseteq I'$ according to Property 1). Hence, $\forall c \in L$, the attributes satisfied by $extent(c)$ in L' include those satisfied by $extent(c)$ in L and others belonging to $M' \setminus M$. Thus, $extent(c)$ is the extent of a concept in L' because it contains the only objects satisfying these attributes (it is closed). Consequently, $\exists h(c) \in L'$ such that $extent(c) = extent(h(c))$ and $intent(c) \subseteq intent(h(c))$ as $h(c)$ may satisfy additional attributes belonging to $M' \setminus M$, but it satisfies at least all those of $intent(c)$. So, $L \preceq L'$. □

5.2.2 The lattice expansion functions E

Instead of considering that $RCA(K^0) = FCA^*(F^\infty(K^0))$, it is possible to consider a function $E : \mathcal{L} \mapsto \mathcal{L}$, such that $RCA(K^0) = F^\infty(FCA^*(K^0))$. In principle, the definition of E amounts to first scaling and then applying FCA, though F does the opposite (see Figure 5.1).

We define $E_{K^0,R,\Omega}$, the lattice expansion function attached to a relational context $\langle K^0, R\rangle$ and a set Ω of scaling operators.

Definition 8 (Lattice expansion function)**.** *Given a relational context $\langle K^0, R\rangle$ and a set of relational scaling operations Ω the function $E_{K^0,R,\Omega} : \mathcal{L}_{K^0,R,\Omega} \mapsto \mathcal{L}_{K^0,R,\Omega}$ is defined by:*

$$E_{K^0,R,\Omega}(L) = FCA^*(\sigma^*_\Omega(\kappa^*(L), R, L))$$

Here again, K^0 is only used to constrain the domain of the function, not its expression. From now on, we will abbreviate $\mathcal{L}_{K^0,R,\Omega}$ as $\mathcal{L}$ and $E_{K^0,R,\Omega}$ as E. In consequence, E is the function corresponding to F in the sense that $E = FCA \circ F \circ \kappa^*$.

E is an extensive and monotone internal operation for $\mathcal{K}$:

Property 9. *$\forall L \in \mathcal{L}$, $E(L) \in \mathcal{L}$*

Proof. $\kappa^*(L) \in \mathcal{K}$. $K = \sigma^*_\Omega(\kappa^*(L), R, L)$ adds attributes from $D_{\Omega,R,N(G)}$ to $\kappa^*(L)$, hence $K \in \mathcal{K}$. Consequently, $E(L) = FCA^*(K) \in \mathcal{L}$. □

Property 10 (E is monotone and extensive)**.** *The function E attached to a relational context and a set of scaling operator satisfies:*

(monotony) $$L \preceq L' \Rightarrow E(L) \preceq E(L')$$

(extensivity) $$L \preceq E(L)$$

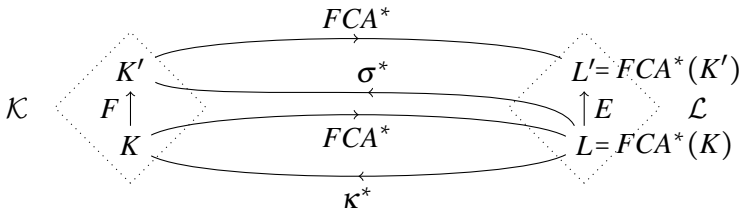


Figure 5.1: Relations between F and E through the alternation of FCA^* and σ^*_Ω.

Proof. monotony $L \preceq L'$ entails that all concepts of L are found in L' with a larger intent. Consequently, $D_{\Omega,R,N(G)}(L) \subseteq D_{\Omega,R,N(G)}(L')$. This entails that $\sigma^*_\Omega(K,R,L')$ extends K with more attributes than $\sigma^*_\Omega(K,R,L)$. Hence $E(L) \preceq E(L')$ because $E(L)$, like L is the application of FCA^* to the same formal context, to which has been added attributes.

extensivity $L = FCA^*(K)$ for $K \in \mathcal{K}$, thus $K \subseteq \sigma^*_\Omega(K,R,L)$. $E(L) = FCA^*(\sigma^*_\Omega(K,R,L))$ so it will have at least all concepts generated by K (identified by extent) because $E(L)$ is closed by 'meet'. Thus, for each concept $c \in L$ there exists $h(c) \in E(L)$ (with $extent(c) = extent(h(c))$ eventually with a larger intent, i.e. $intent(c) \subseteq intent(h(c))$), generated by the new scaled attrubutes. Hence, $L \preceq E(L)$. □

Monotony is also called order-preservation. It corresponds to the non-(intent-)contracting concept property of [Rouane-Hacene et al. 2013b].

5.2.3 Fixed points of E

Given E, it is possible to define its set of fixed points, i.e. the sets of concept lattices closed for E, as:

Definition 9 (fixed point)**.** *A concept lattice $L \in \mathcal{L}$ is a fixed point for a lattice expansion function E, if $E(L) \simeq L$. We call $fp(E)$ the set of fixed points for E.*

We can define:

$$lfp(E) = \bigwedge_{L \in fp(E)} L \text{ and } gfp(E) = \bigvee_{L \in fp(E)} L$$

Since $\mathcal{L}$ is a complete lattice and E is order-preserving (or monotone) on $\mathcal{L}$, then we can apply the Knaster-Tarski theorem. This warrants that there exists least and greatest fixed points of E in $\mathcal{L}$.

5.3 Well-grounded and least fixed-point semantics

RCA may be redefined as

$$RCA_\Omega(K^0,R) = FCA^*(F^\infty(K^0))$$

i.e. RCA iterates F from K^0 until closure, and ultimately applies FCA^*. Alternatively, RCA may be redefined as

$$RCA_\Omega(K^0,R) = E^\infty(FCA^*(K^0))$$

i.e. RCA iterates E from $FCA^*(K^0)$ until closure.

The definition of E first applies scaling and then FCA, though F does the opposite. In consequence, E is the function corresponding to F in the sense that $FCA^* \circ E = F \circ FCA^*$ (see Figure 5.1). It seems thus that RCA returns a fixed point of E. Hence the question: which fixed point is returned by RCA's well-grounded semantics?

5.3.1 The RCA well-grounded semantics is the least fixed-point semantics

Since K^0 belongs to $\mathcal{K}$ and $FCA(K^0)$ belongs to $\mathcal{L}$, then RCA is indeed based on E and F fixed points. These are the least fixed points.

Proposition 11 (The RCA algorithm computes the least fixed point)**.** *Given F the context expansion function and E the lattice expansion function associated to K^0, R and Ω,*

$$RCA_\Omega(K^0, R) = FCA^*(lfp(F_{K^0,R,\Omega}))$$

and

$$RCA_\Omega(K^0, R) = lfp(E_{K^0,R,\Omega})$$

Proof. Concerning the first equation, $RCA_\Omega(K^0, R) = FCA^*(F^n(K^0))$ for some n at which $F(F^n(K^0)) = F^n(K^0)$ [Rouane-Hacene et al. 2013a]. Let $K^\infty = F^n(K^0)$, $K^\infty \in fp(F)$ (Definition 9). $\forall K \in fp(F)$, $K \in \mathcal{K}$, thus $K^0 \subseteq K$ because all the contexts in $\mathcal{K}$ contain M^0. By monotony (Property 5), $K^\infty = F^n(K^0) \subseteq F^n(K) = K$, because K is a fixed point. Thus, K^∞ is a fixed point more specific than all fixed points: it is the least fixed point.

Concerning the second equation, $RCA_\Omega(K^0, R) = E^n(K^0)$ for some n at which $E(E^n(K^0)) = E^n(K^0))$ [Rouane-Hacene et al. 2013a]. $E(K^0) \in \mathcal{L}$, hence (by Property 9), $E^n(K^0) \in \mathcal{L}$. Moreover, $E(E^n(K^0)) = E^n(K^0))$ thus $E^n(K^0)) \in fp(E)$. In addition, $\forall L \in fp(E)$, $E(K^0) \preceq L$ because the context from which L is created contains at least all attributes of K^0. But if $E^i(K^0) \preceq L$, then $E^{i+1}(K^0) \preceq L$ because by monotony (Property 10), $E^{i+1}(K^0) = E(E^i(K^0)) \preceq E(L)$ and E is idempotent on fixed points (by Definition 9). Thus, $RCA_\Omega(K^0, R)$ is a fixed point more specific than all fixed points: it is the least fixed point. □

5.3.2 Greatest fixed point

A natural question is how to obtain the greatest fixed point. In fact, under this approach this is (theoretically) surprisingly easy.

Proposition 12. $gfp(F_{\langle G,M^0,I^0\rangle,R,\Omega}) = K^{\langle R,N(G)\rangle}_{+D_{\Omega,R,N(G)}}(\langle G, M^0, I^0\rangle)$

Proof. This context is the greatest element of $\mathcal{K}$ as it contains all attributes of $D_{\Omega,R,N(G)}$. It is also a fixed point because F is extensive and internal. □

The lattice corresponding to the greatest fixed point will be $L = FCA^*(gfp(F_{K^0,R,\Omega}))$.

This result is easy but very uncomfortable. The obtained lattice may contain many useless attributes. Indeed, $\exists r.c$ is well defined by the incidence relation, but it is of no use to RCA if c does not belong to L.

For instance, in the example of Section 4.2, the attribute $\exists p.A$ belongs to $D_{\Omega,R,N(G)}$ though A does not belong to the maximal lattice L_3, because it is not closed. The fact that both a and b satisfy this attribute makes that it will find its place in the intent of AB. If one considers the lattice in isolation, this is perfectly valid because the scaled context is well-defined: $\exists p.A$ is just an attribute among others satisfied by a and b. However, if the lattice is transformed in a description logic TBox, this is not correct to refer to an undefined class (here A).

On the contrary, there exist problems, such as the one provided in §4.3, in which the greatest fixed point is the powerset lattice, i.e. in which all attributes are supported, and the least fixed point is directly $FCA^*(K^0)$.

This problem is even more embarrassing if one wants to enumerate all fixed points, which are as many solutions to the RCA problem: many of these will feature such non-supported attributes.

5.4 Self-supported fixed points

In order to define more interesting results for RCA we introduce the notion of support, and more specifically in RCA^0 of self-support. It specifies that a concept lattice is supported if its intents only refer to concepts in this lattice. We describe a function Q which suppresses non supported attributes and whose closure provides self-supported lattices. We then identify the interesting results as self-supported fixed points.

The problem is that both F and E are extensive functions. Hence, it is possible, starting from anywhere in $\mathcal{K}$ or $\mathcal{L}$, to consider attributes that do not refer to concepts and these attributes will be preserved. As a consequence, there are fixed points with these unwanted attributes and they are also found in the greatest fixed point. This is not the result that we expect: we need the results to be self-supported.

One may consider identifying such attributes from the greatest fixed point and forbidding them. However, these meaningless attributes are contextual: one supported attribute in the greatest fixed point, may be non supported in a smaller lattice. This is a difficulty for enumerating these fixed points.

Instead, we consider only self-supported lattices, i.e. lattices whose intents only refer to their own concepts.

Definition 10 (Self-supported lattices)**.** *A concept lattices L is* self-supported *if* $\forall c \in L, intent(c) \subseteq D_{\Omega,R,L}$.

The set of interesting lattices that may be returned by RCA^0 can be circumbscribed as the self-supported fixed points of E. Such lattices are both stable and self-supported. Moreover, by construction of $\mathcal{K}$ and $\mathcal{L}$, they cover K^0, i.e. they contain all attributes in M^0.

But the definition of self-supported lattices does not provide a direct way to transform a non self-supported lattice into a self-supported one. Simply suppressing non-supported attributes from intents could result in non concepts (with non-closed extents). One possible way to solve this problem consists of extracting only the attributes currently in the lattice and applying FCA^* to the resulting context.

For that purpose, we introduce a filtering or purging function $\pi : \mathcal{L} \mapsto \mathcal{K}$ which suppresses *from the induced context* ($\kappa(L)$) those attributes non supported *by the lattice*:

$$\pi(L) = K^{\langle R,L\rangle}_{-D_{\Omega,R,N(G)\setminus L}}(\kappa(L))$$

such that $\kappa(L) = \langle G,M,I\rangle$. This can be generalised for RCA and as for σ_Ω and FCA, it is possible to introduce π^*:

$$\pi(L,\{L_x\}_{x\in X}) = K^{\langle R,\{L_x\}_{x\in X}\rangle}_{-\cup^{\varsigma\in\Omega}_{r\in R|r\subseteq G\times G_z} D_{\varsigma,r,N(G)\setminus L_z}}(\kappa^*(L)$$
$$\pi^*(\{L_x\}_{x\in X}) = \{\pi(L_x,\{L_z\}_{z\in X})\}_{x\in X}$$

One can define $Q : \mathcal{L} \mapsto \mathcal{L}$, such that

$$Q(L) = FCA^*(\pi^*(L))$$

or $P : \mathcal{K} \mapsto \mathcal{K}$, such that $P(K) = \pi^*(FCA^*(K))$, see Figure 5.2.

Contrary to E, Q is anti-extensive and monotone:

Property 13 (Q is anti-extensive and monotone)**.** *The function Q satisfies:*

(anti-extensivity) $\quad Q(L) \preceq L$

(monotony) $\quad L \preceq L' \Rightarrow Q(L) \preceq Q(L')$

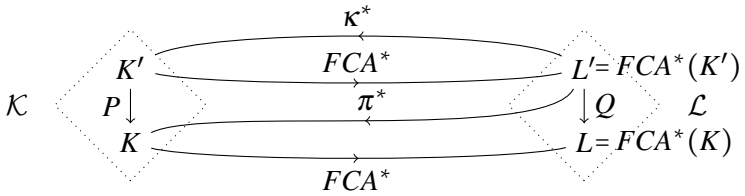


Figure 5.2: Relations between P and Q through the alternation of FCA^* and π^*.

Proof. anti-extensivity $\pi^*(L) \subseteq \kappa^*(L)$ because π^* simply suppresses attributes from $\kappa^*(L)$. Hence, $FCA^*(\pi^*(L)) \preceq FCA^*(\kappa^*(L))$ because the latter contain all concepts of the former (identified by extent) eventually featuring the removed attributes. Moreover, $FCA^*(\kappa^*(L)) = L$ by definition, thus $Q(L) = FCA^*(\pi^*(L)) \preceq FCA^*(\kappa^*(L)) = L$.

monotony If $L \preceq L'$, then $\kappa^*(L) \subseteq \kappa^*(L')$, otherwise FCA^* would not generate a smaller lattice. In addition, $L \preceq L'$ entails $N(G) \setminus L \supseteq N(G) \setminus L'$ which entails $D_{\Omega,R,N(G)\setminus L} \supseteq D_{\Omega,R,N(G)\setminus L'}$, which finally together leads to $M \setminus D_{\Omega,R,N(G)\setminus L} \subseteq M' \setminus D_{\Omega,R,N(G)\setminus L'}$. Then, $\pi^*(L) \subseteq \pi^*(L')$ because a smaller context supported by a smaller lattice cannot result in a larger context. Hence, $Q(L) = FCA^*(\pi^*(L)) \preceq FCA^*(\pi^*(L')) = Q(L')$. □

Like E, Q is not a closure operator as it is not idempotent. However, with the same arguments as [Rouane-Hacene et al. 2013a], it can be argued that the repeated application of Q converges to a self-supported concept lattice.

Property 14. $\forall L \in \mathcal{L}, \exists n;\ Q^n(L) = Q^{n+1}(L)$.

Proof. First, L is a finite concept lattice. Moreover, $Q(L) \preceq L$, hence it not possible to build an infinite chain of non converging application of Q since at each iteration, either π^* suppresses no attribute (and then closure has been reached), or it suppresses at least one attribute and then a strictly smaller context is reached. Ultimately, the least fixed point $lfp(Q) = FCA^*(K^0)$ is reached. It is a fixed point because $\kappa^*(FCA^*(K^0)) = K^0$ contains no scaled attribute. □

By convention, we note Q^∞ the closure function associated with Q and $fp(Q)$, the set of fixed points of Q. Like with E, it is possible to apply the Knaster-Tarski theorem to show that $\langle fp(Q), \preceq \rangle$ is a complete lattice.

The fixed points of Q are exactly those self-supported lattices in $\mathcal{L}$:

Property 15. *For any $L \in \mathcal{L}$, L is self-supported iff $L \in fp(Q)$.*

Proof. Any fixed point for Q is self-supported because if $Q(L) = L$, this is because π^* does not find any non-supported attribute in the lattice intents. This means that all of them are supported by L. Conversely, each self-supported lattice $L \in \mathcal{L}$ is such that $\pi^*(L) = \kappa^*(L)$ because all concepts of L only refer to attributes of L, so π^* does not suppress any attribute from the context. Thus, $Q(L) = FCA^*(\pi^*(L)) = FCA^*(\kappa^*(L)) = L$ (by construction of κ^*), hence $L \in fp(Q)$. □

To complete the description of Q, it is possible to establish that its least fixed point is $FCA^*(K_0)$.

Property 16. $lfp(Q) = FCA^*(K_0)$.

Proof. $\kappa^*(FCA^*(K_0)) = K_0$ hence $\pi^*(FCA^*(K_0)) = K_0$ because, it is not possible to suppress attributes from K_0 which being a formal context does not refer to any concept (and in RCA^0 this set of attributes is reduced to $\varnothing$). Thus, $Q(FCA^*(K_0)) = FCA^*(\pi^*(FCA^*(K_0))) = FCA^*(K_0)$. Moreover, $\forall L \in \mathcal{L}$, $FCA^*(K_0) \preceq L$. Hence, $FCA^*(K_0)$ is a fixed point of Q and all other fixed points are greater. □

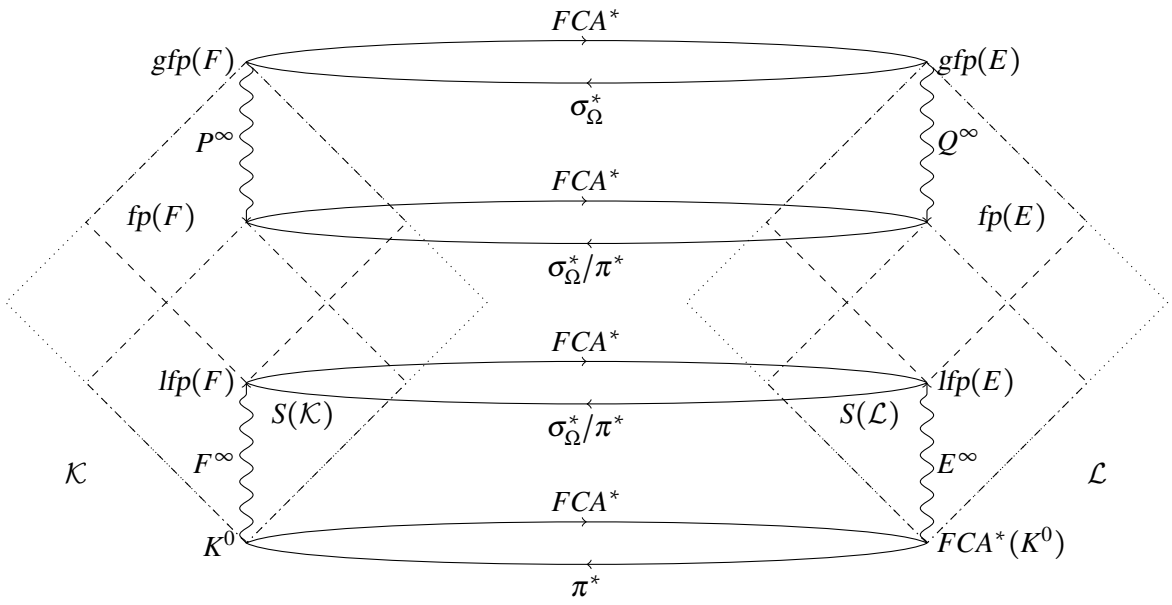


Figure 5.3: The $\mathcal{L}$ (resp. $\mathcal{K}$) lattice and effects of E and Q (resp. F and P) for characterising $fp(E)$ and $S(\mathcal{L})$ (resp. $fp(F)$ and $S(\mathcal{K})$).

We end up with two operations, E and Q, the former extensive and the latter anti-extensive. If we consider concept lattices from the standpoint of the extents, Q decreases the set of concepts of a lattice and E increases them.

An interesting property of the functions Q and E is that they preserve each other stability:

Property 17 (Q is internal to $fp(E)$)**.** $\forall L \in fp(E)$, $Q(L) \in fp(E)$

Proof. If $L \in fp(E)$, this means that $E(L) = L$ and, in particular, that σ^*_Ω does not scale new attributes based on the concepts in L. $Q(L) \preceq L$, so that $Q(L)$ contains not more concepts than L. $Q(L)$ having not more concepts than L, σ^*_Ω cannot scale new attributes either ($\sigma^*_\Omega(Q(L)) \subseteq \sigma^*_\Omega(L) = \varnothing$). Hence, $Q(L) \in fp(E)$. □

E has the advantage of preserving self-supportivity.

Property 18 (E is internal to $fp(Q)$)**.** $\forall L \in fp(Q)$, $E(L) \in fp(Q)$.

Proof. If $L \in fp(Q)$, all attributes in intents of L are supported by concepts in L (Property 15). $L \preceq E(L)$, so these concepts are still in $E(L)$. Moreover, $E = FCA^* \circ \sigma^*_\Omega$ and σ^*_Ω first adds to $\kappa^*(L)$ attributes which are supported by L. Hence, the attributes in $\kappa^*(L)$ and those scaled by σ^*_Ω are still supported by $E(L)$. □

In addition, the closure operations associated with the two functions preserve its extrema.

Property 19. *$Q^\infty(gfp(E)) = gfp(Q)$ and $E^\infty(lfp(Q)) = lfp(E)$*

Proof. $\forall L \in \mathcal{L}$, $L \preceq gfp(E)$ (from Proposition 12) and Q and thus Q^∞ is order preserving (Property 13), hence $Q^\infty(L) \preceq Q^\infty(gfp(E))$. Moreover, $Q^\infty(gfp(E)) \in fp(Q)$, thus $Q^\infty(gfp(E)) = gfp(Q)$.

Similarly, $\forall L \in \mathcal{L}$, $lfp(Q) \preceq L$ (Property 16) and E and thus E^∞ is order preserving (Property 10), hence $E^\infty(lfp(Q)) \preceq E^\infty(L)$. Moreover, $E^\infty(lfp(Q)) \in fp(E)$, thus $E^\infty(lfp(Q)) = lfp(E)$. □

The interesting solutions for RCA are the self-supported fixed points of E, or said otherwise, the elements of $fp(E) \cup fp(Q)$.

These functions are instrumental to provide the infimum and supremum of our desired lattices (see also Figure 5.3):

Proposition 20. $\forall L \in fp(E) \cap fp(Q)$, $lfp(E) \preceq L \preceq gfp(Q)$.

Proof. $lfp(E)$ is the lower bound for $fp(E)$. Assume that $lfp(E) \notin fp(Q)$, then there would exist $Q^\infty(lfp(E)) \in fp(Q)$ (by Property 14). By Property 17, $Q^\infty(lfp(E)) \in fp(E)$ and due to Property 13 (anti-extensivity), $Q^\infty(lfp(E)) \preceq lfp(E)$. This contradicts that $lfp(E)$ is the lower bound for $fp(E)$. Hence, $lfp(E) \in fp(E) \cap fp(Q)$ and is its infimum.

Similarly, $gfp(G)$ is the upper bound for $fp(Q)$. Assume that $gfp(G) \notin fp(E)$, then there would exist $E^\infty(gfp(G)) \in fp(E)$ [Rouane-Hacene et al. 2013a]. By Property 18, $E^\infty(gfp(G)) \in fp(Q)$ and due to Property 10 (extensivity), $gfp(G) \preceq E^\infty(gfp(G))$. This would mean that $gfp(G)$ is not the upper bound for $fp(Q)$. Hence, $gfp(Q) \in fp(E) \cap fp(Q)$ and is its supremum. □

The elements of $fp(E) \cap fp(Q)$ thus belong to the interval sublattice $[lfp(E)\ gfp(Q)]$. However they do not cover it. The converse of Proposition 20 does not hold in general as shown by the example of Section 4. Indeed L_3 is $lfp(E)$ and not all sublattices of L_3 are solutions. For instance, the lattice of Figure 5.6 belongs to the interval, but not to $fp(E) \cap fp(Q)$.

5.5 Structure of fixed points

Thus we are interested by characterising better the elements of $fp(E) \cap fp(Q)$. Figure 5.4 shows how this set can be found in our depiction of $\mathcal{L}$.

We use the closure of E and Q, E^∞ and Q^∞ to provide functions which maps elements of $\mathcal{L}$ into a stable lattice in $fp(E) \cap fp(Q)$. Both are closure operators.

We show that their elements can be obtained from any element of $\mathcal{L}$ by applying the two closure operators Q^∞ and E^∞ in a row.

Property 21. *$Q^\infty \circ E^\infty$ (resp. $E^\infty \circ Q^\infty$) is order-preserving and idempotent:*

(order-preservation) $\quad \forall L, L' \in \mathcal{L}, L \preceq L' \Rightarrow Q^\infty \circ E^\infty(L) \preceq Q^\infty \circ E^\infty(L')$

(idempotence) $\quad Q^\infty \circ E^\infty \circ Q^\infty \circ E^\infty(L) = Q^\infty \circ E^\infty(L)$

Proof. We prove it for $Q^\infty \circ E^\infty$, the $E^\infty \circ Q^\infty$ case is strictly dual.

order-preservation is obtained as the combination of order-preservation of the two others: $L \preceq L'$, hence $E(L) \preceq E(L')$ (Property 10) and also $E^\infty(L) \preceq E^\infty(L')$, thus $Q^\infty \circ E^\infty(L) \preceq Q^\infty \circ E^\infty(L')$ (Property 13).

idempotence is obtained from Property 22: $\forall L \in \mathcal{L}$, $Q^\infty \circ E^\infty(L) \in fp(E) \cap fp(Q)$, hence $Q^\infty \circ E^\infty(L) = L$ and $Q^\infty \circ E^\infty(L) \circ Q^\infty \circ E^\infty(L) = L$, thus $Q^\infty \circ E^\infty \circ Q^\infty \circ E^\infty(L) = Q^\infty \circ E^\infty(L)$. □

It does not seems that these functions are neither extensive nor anti-extensive (see Figure 5.5 and eventually the example of Figure 5.6 and 5.7). Hence, they would not be closure operators.

The set of self-supported fixed points are those elements in the image of $\mathcal{L}$ by the composition of these two closure operations, in any order.

Property 22. $Im(Q^\infty \circ E^\infty) = fp(E) \cap fp(Q) = Im(E^\infty \circ Q^\infty)$

Proof. We show it for $Q^\infty \circ E^\infty$, the other part is dual:

$\subseteq$ By definition, $Im(Q^\infty \circ E^\infty) \subseteq Im(Q^\infty) = fp(Q)$. Moreover, $Im(E^\infty) = fp(E)$, but by Property 17, if $L \in fp(E)$, then $Q^\infty(L) \in fp(E)$. Hence, $Im(Q^\infty \circ E^\infty) \subseteq fp(E) \cap fp(Q)$.

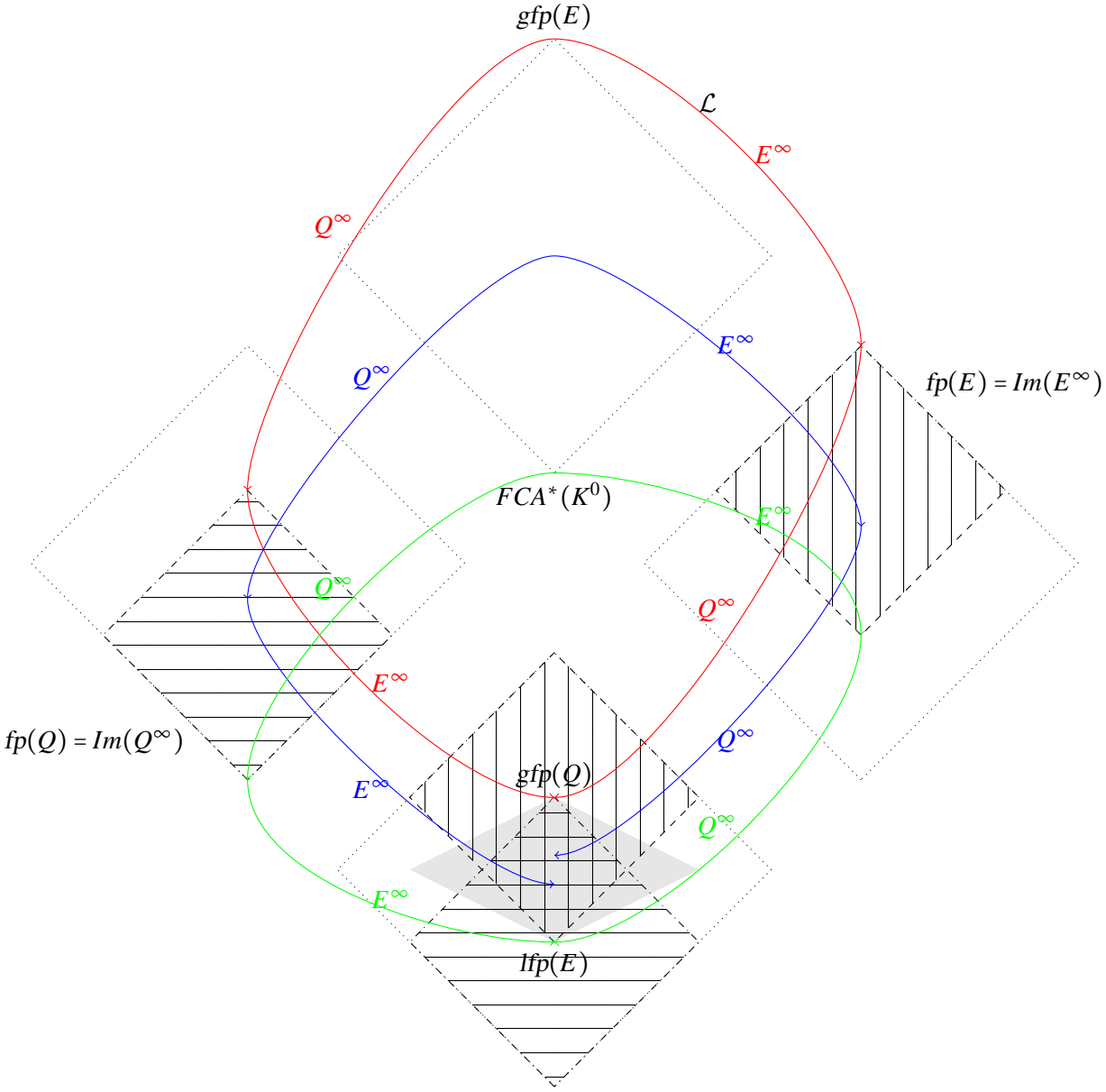


Figure 5.4: Illustration of Properties 19, 20, 21, 22, 23, 25 and 26. The figure displays four times $\mathcal{L}$ and the images of $gfp(E)$ (red), a random lattice (blue) and $lfp(Q) = FCA^*(K^0)$ (green) through Q^∞ (left) and E^∞ (right). $fp(E)$ is drawn in vertical lines; $fp(Q)$ in horizontal lines and the grey area depicts the interval $[lfp(E)\ gfp(Q)]$.

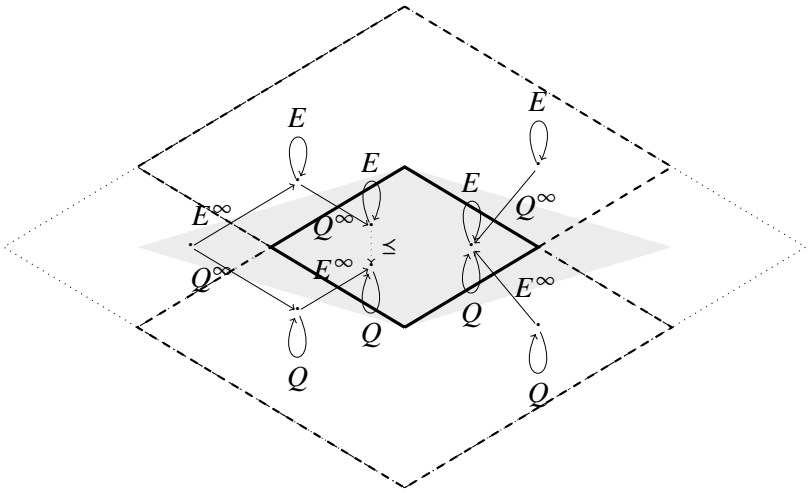


Figure 5.5: Illustration of the convergence of $Q^\infty \circ E^\infty$ and $E^\infty \circ Q^\infty$ depending on their original position (in dotted, $\mathcal{L}$, in dashed $fp(E) \cup fp(Q)$, in plain $fp(E) \cap fp(Q)$).

$\supseteq$ $\forall L \in fp(Q) \cap fp(E)$, $L \in fp(E)$, thus $E^\infty(L) = L \in fp(Q)$, hence $Q^\infty(L) = L$. Thus $L \in Im(Q^\infty \circ E^\infty)$, hence $fp(Q) \cap fp(E) \subseteq Im(Q^\infty \circ E^\infty)$. □

The fact that the image of these functions is in their fixed points is a good news, because it means that applying one of these function to whatever lattice $L \in \mathcal{L}$ converges to $fp(E) \cap fp(Q)$.

So the structure of $fp(E) \cap fp(Q)$ is that of a complete lattice:

Property 23. *$\langle fp(E) \cap fp(Q), \preceq \rangle$ is a complete sublattice of $\langle \mathcal{L}, \preceq \rangle$.*

Proof. $fp(E) \cap fp(Q) = Im(Q^\infty \circ E^\infty)$ (Property 22) and $Im(Q^\infty \circ E^\infty) = fp(Q^\infty \circ E^\infty)$ due to idempotence (Property 21), hence the Knaster-Tarski theorem can be applied based on Property 21 (order-preservation), concluding that it is a complete lattice. It is obviously included in $\mathcal{L}$, thus this is a sublattice of $\langle \mathcal{L}, \preceq \rangle$. □

For any lattice within the fixed points, i.e. $fp(E)$ *or* $fp(Q)$, the two functions are equal.

Property 24. $\forall L \in fp(E) \cup fp(Q), Q^\infty \circ E^\infty(L) = E^\infty \circ Q^\infty(L)$

Proof. These can be found on Figure 5.5. For any lattice L belonging to $fp(E) \cap fp(Q)$ (the gray area), $Q(L) = E(L) = L$, hence $Q^\infty \circ E^\infty(L) = E^\infty \circ Q^\infty(L) = L$. Similarly, for any lattice L belonging to $fp(E)$, then $E^\infty(L) = L$, so $Q^\infty \circ E^\infty(L) = Q^\infty(L)$. However, by Property 18, since $L \in fp(E)$, so $Q^\infty(L) \in fp(E)$. This means that $E^\infty \circ Q^\infty(L) = Q^\infty(L)$ as well. The same can be proved for $L \in fp(Q)$ with Property 17. □

In particular, this applies to the bounds of $fp(E) \cap fp(Q)$:

Property 25. $Q^\infty \circ E^\infty(gfp(E)) = E^\infty \circ Q^\infty(gfp(E)) = gfp(Q)$ *and* $Q^\infty \circ E^\infty(lfp(Q)) = E^\infty \circ Q^\infty(lfp(Q)) = lfp(E)$

Independent proof. For the first term of the first equation $E^\infty(gfp(E)) = gfp(E)$, and then by Property 19(a), $Q^\infty \circ E^\infty(gfp(E)) = gfp(Q)$, moreover for the second term by Property 19(a) $Q^\infty(gfp(E)) = gfp(Q)$ and then by Property 17, $gfp(Q) \in fp(E)$, hence $E^\infty(gfp(Q)) = gfp(Q)$, thus $E^\infty \circ Q^\infty(gfp(E)) = gfp(Q)$.

Similarly, the second equation relies on Property 19(b). For the first term by Property 19(b) $E^\infty(lfp(Q)) = lfp(E)$ and then by Properties 18 and 15, $lfp(E) \in fp(Q)$, hence $Q^\infty(lfp(E)) = lfp(E)$, thus $Q^\infty \circ$

	$\exists p.AB$	$\exists p.ABD$	$\exists p.C$	$\exists p.D$	$\exists p.ABCD$
a	×	×			×
b	×	×			×
c		×		×	×
d			×		×

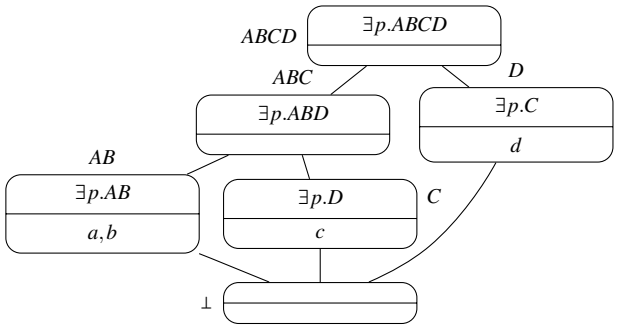


Figure 5.6: Counter-example to the conjecture: the context $K^{\#}$ (left) and the corresponding concept lattice $L^{\#} = FCA^{*}(K^{\#})$ (right).

	$\exists p.AB$	$\exists p.ABD$	$\exists p.ABC$	$\exists p.C$	$\exists p.D$	$\exists p.ABCD$
a	×	×	×			×
b	×	×	×			×
c		×			×	×
d			×	×		×

	$\exists p.AB$	$\exists p.C$	$\exists p.D$	$\exists p.ABCD$
a	×			×
b	×			×
c			×	×
d		×		×

Table 5.1: Contexts obtained by $\sigma^{*}_{\exists}(L^{\#})$ and $\pi^{*}(L^{\#})$.

$E^{\infty}(lfp(Q)) = lfp(E)$. Moreover, for the second term $Q^{\infty}(lfp(Q)) = lfp(Q)$, and then by Property 19(b), $E^{\infty} \circ Q^{\infty}(lfp(Q)) = lfp(E)$. □

However, this does not generally holds as illustrated by Example 2.

Example 2 (Counterexample to equality)**.** *Consider the example given in Figure 5.6. $K^{\#} \in \mathcal{K}$ because all attributes belong to $D_{\Omega,R,N(G)}$. FCA^{*} generates the corresponding lattice $L^{\#} \in \mathcal{L}$. In fact, $L^{\#} \in [lfp(E)\ gfp(Q)] \setminus (fp(E) \cap fp(Q))$. Applying scaling or purging will provide the two contexts of Table 5.1 which correspond to the two lattices of Figure 5.7 ($E(L^{\#})$ and $Q(L^{\#})$). These two lattices belong to $fp(E) \cap fp(Q)$, hence they are in full closed form: $E(L^{\#}) = Q^{\infty} \circ E^{\infty}(L^{\#})$ and $Q(L^{\#}) = E^{\infty} \circ Q^{\infty}(L^{\#})$. Yet they are not isomorphic... What can be said, in this case, is that $E^{\infty} \circ Q^{\infty}(L^{\#}) \preceq Q^{\infty} \circ E^{\infty}(L^{\#})$. This is the result of σ eventually adding needed support and π suppressing unsupported attributes.*

It seems correct that when Q is first applied, it suppresses non-supported attributes which cannot be recovered by scaling. On the other side, E applied first may scale attributes which may support previously non-supported attributes (*ABC* in Example 2). These will not be suppressed any more.

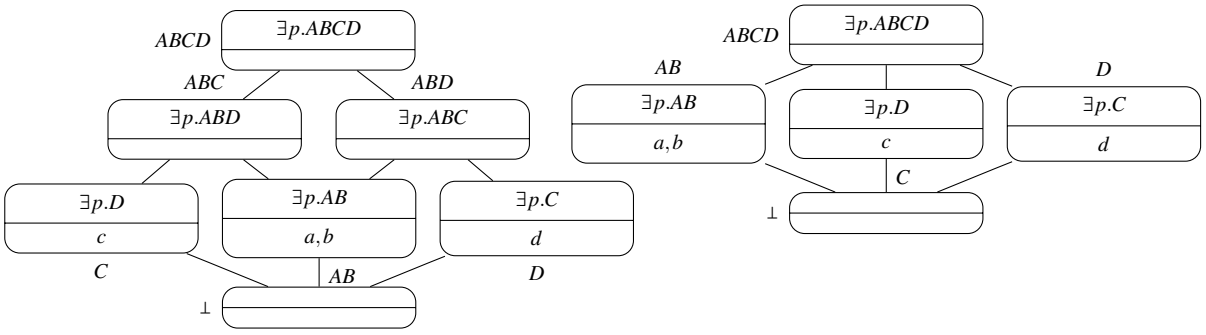


Figure 5.7: Lattices obtained from the contexts of Table 5.1 by $E(L^{\#}) = FCA^{*}(\sigma^{*}_{\exists}(L^{\#}))$ and $Q(L^{\#}) = FCA^{*}(\pi^{*}(L^{\#}))$.

What is shown by Property 26, is that, in addition there is still a homomorphism between the two resulting lattices.

Property 26. $\forall L \in \mathcal{L}, E^\infty \circ Q^\infty(L) \preceq Q^\infty \circ E^\infty(L)$

Proof. $Q(L) \preceq L$ entails $Q^\infty(L) \preceq L$. But E is monotonous, so is E^∞, hence $E^\infty \circ Q^\infty(L) \preceq E^\infty(L)$. Q is also monotonous, thus $Q^\infty \circ E^\infty \circ Q^\infty(L) \preceq Q^\infty \circ E^\infty(L)$. However, $Q^\infty \circ E^\infty(L) \in fp(E) \cup fp(Q)$ so $Q^\infty \circ E^\infty \circ Q^\infty(L) = E^\infty \circ Q^\infty(L)$. This means that $E^\infty \circ Q^\infty(L) \preceq Q^\infty \circ E^\infty(L)$. □

Alternative proof. The same reasoning can be held from $L \preceq E(L)$. This entails $L \preceq E^\infty(L)$. E and Q being monotonous still entails that E^∞ and Q^∞ are. Hence, $Q^\infty(L) \preceq Q^\infty \circ E^\infty(L)$ and $E^\infty \circ Q^\infty(L) \preceq E^\infty \circ Q^\infty \circ E^\infty(L)$. But, $Q^\infty \circ E^\infty(L) \in fp(E) \cup fp(Q)$, thus $E^\infty \circ Q^\infty \circ E^\infty(L) = Q^\infty \circ E^\infty(L)$. Thus, $E^\infty \circ Q^\infty(L) \preceq Q^\infty \circ E^\infty(L)$. □

It is thus unclear what to do with $E^\infty \circ Q^\infty$ and $Q^\infty \circ E^\infty$ in general. For instance, if one needs an operation to map elements of $\mathcal{L}$ to $fp(E) \cap fp(Q)$, which one is preferable? There may be an interest in studying the interval $[E^\infty \circ Q^\infty(L), Q^\infty \circ E^\infty(L)]$. Does it contain only fixed points? Are these the image of other lattices?

Similar questions may be raised with respect to the use of $(E \circ Q)^\infty$ and $(Q \circ E)^\infty$, and first of all if these are well defined.

It is also possible to consider the congruence induced by these functions. Indeed, $Q^\infty \circ E^\infty$ or $E^\infty \circ Q^\infty$ may be considered as a lattice homorphisms in $\mathcal{L}$. The equivalence relation they induce is defined by: $ker\, h = \{\langle x, y\rangle \in \mathcal{L} \times \mathcal{L} \mid h(x) = h(y)\}$. Let us call $\mathcal{L}|_{ker\, Q^\infty \circ E^\infty}$ the subset of $\mathcal{L}$ containing the common image of each class in $ker\, Q^\infty \circ E^\infty$. $\mathcal{L}|_{ker\, Q^\infty \circ E^\infty}$ is exactly $Im(Q^\infty \circ E^\infty)$. The same holds for $\mathcal{L}|_{ker\, E^\infty \circ Q^\infty} = Im(E^\infty \circ Q^\infty)$.

5.6 FCA and hierarchical RCA

In FCA, a single concept lattice corresponds to a context. All support is provided in this context and there is no room to introduce new concept without breaking closedness. Moreover, FCA is external to the kind of fixed points that we are considering.

Subsequently in RCA without circular dependencies, or hierarchical RCA (HRCA), all contexts may be organised in strata so that a context belongs to the stratum above all those of the contexts it depends on. Contexts of the first stratum only rely on FCA since $R = \varnothing$. They thus only have a single fixed point (actually, there is no scaling so $\mathcal{G}$ is reduced to one single context, K^0). Contexts of each strata can be processed once for all when the previous strata have been processed[2]. Since the contexts do not depend on other contexts of the same stratum, it is not necessary to run FCA^* before scaling the context. Scaling on Stratum i will occur once to add the new attributes depending on Ω, R and the lattices of the previous strata ($L_{<i}$). But since these will not change any more, further scaling will not bring new attributes (Property 27). It is thus a single FCA performed in this scaled context.

Property 27. *In HRCA, $E = E^\infty$ and $Q = Q^\infty$*

Proof. E performs σ^*_Ω and then FCA^*. In hierarchical RCA, σ^*_Ω adds to $\kappa^*(L)$ the attributes of $D_{\Omega,R,L_{<i}}$. Applying E once again, would not change the result since, there will be no attributes to add because Ω and R do not change and $L_{<i}$ are not recomputed. Hence, $E(E(L)) = E(L)$.

Similarly, Q performs π^* and then FCA^*. In hierarchical RCA, π^* removes from $\kappa^*(L)$ the attributes of the range of relations that do not belong to $D_{\Omega,R,L_{<i}}$. Applying Q once again, would not change the result since, there will be no more attributes to remove because Ω and R do not change and $L_{<i}$ are not recomputed. Hence, $Q(Q(L)) = Q(L)$. □

[2] This differs from the classical RCA process, but the result will be strictly equivalent.

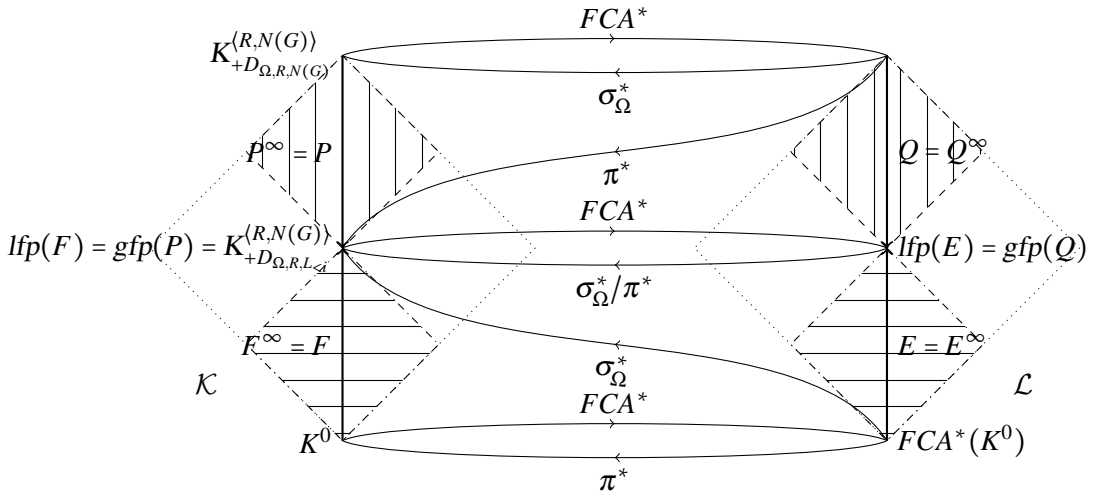


Figure 5.8: Hierarchical RCA. All functions are their own closures and they return the same fixed point ($lfp(E) = gfp(Q)$).

Moreover, the least fixed point of E will be the same as the greatest fixed point of Q (Property 28). Hence, there is only one self-supported fixed point.

Property 28. *In HRCA, $gfp(Q) = lfp(E)$.*

Proof. First, all contexts of $\mathcal{K}$ contains the attributes of K^0. $lfp(E) = E^\infty(FCA^*(K^0)) = E(FCA^*(K^0))$ and $gfp(Q) = Q^\infty(FCA^*(K^{\langle R,N(G_{<i})\rangle}_{D_{\Omega,R,N(G_{<i}}}(K^0))) = Q(FCA^*(K^{\langle R,N(G_{<i})\rangle}_{D_{\Omega,R,N(G_{<i}}}(K^0)))$ (Property 27). $E = FCA^* \circ \sigma^*_\Omega$ and $Q = FCA^* \circ \pi^*$. σ^*_Ω adds to K^0 all the attributes of $D_{\Omega,R,L_{<i}}$ and π^* removes from $D_{\Omega,R,N(G_{<i}}$ all the attributes *not* in $D_{\Omega,R,L_{<i}}$ (and $D_{\Omega,R,L_{<i}} \subseteq D_{\Omega,R,N(G_{<i}}$). Thus, $\sigma^*_\Omega(FCA^*(K^0)) = \pi^*(FCA^*(K^{\langle R,N(G_{<i})\rangle}_{D_{\Omega,R,N(G_{<i}}}(K^0)))$ and $lfp(E) = FCA^*(\sigma^*_\Omega(FCA^*(K^0))) = FCA^*(\pi^*(FCA^*(K^{\langle R,N(G_{<i})\rangle}_{D_{\Omega,R,N(G_{<i}}}(K^0)))) = gfp(Q)$. □

A lattice $L \in \mathcal{L}$ is in one of the following cases:

- $L = lfp(E) = gfp(Q)$.
- $lfp(E) \prec L$, which when purged, suppress all attributes in $D_{\Omega,R,N(G)} \setminus D_{\Omega,R,L_{<i}}$, so that $Q(L) = gfp(Q)$.
- $L \prec gfp(Q)$ which, when scaled, receive all attributes in $D_{\Omega,R,L_{<i}}$, so that $E(L) = lfp(E)$.
- $L \not\succeq gfp(Q) \wedge lfp(E) \not\preceq L$, non comparable to $lfp(E)$ nor $gfp(Q)$, these lattices are obtained from contexts which do not contain all attributes of $D_{\Omega,R,L_{<i}}$ and do contain attributes of $D_{\Omega,R,N(G)} \setminus D_{\Omega,R,L_{<i}}$. For these scaling will provide them with all $D_{\Omega,R,L_{<i}}$ and the purging will suppress those elements of $D_{\Omega,R,N(G)} \setminus D_{\Omega,R,L_{<i}}$. In fact, $E \circ Q(L) = Q \circ E(L) = lfp(E)$.

The arguments provided in this section apply to RCA and not to RCA^0 (by definition, a hierarchical RCA^0 context is a FCA context without attribute which is not particularly appealing). The notion of support has replaced that of self-support.

Even with RCA and circular dependencies (between the objects or between the contexts), in many cases, there is only one supported fixed point which is the one computed by the RCA algorithm. But the example of Section 4 shows that, even in RCA^0, there may be several non isomorphic fixed points for E and F.

Conclusion

We determined the dual spaces, of contexts and lattices, in which RCA evolves. We identified the functions, F and E, at the core of RCA and determined that RCA computes the least fixed points of function E.

This result does not mean that RCA is wrong. In FCA, conceptual scaling has been considered as a human-driven analysis tool: a knowledgeable person could provide attribute in this language for describing better the data to be analysed. In RCA, scaling is used as an extraction tool, with the drawback to potentially generate many attributes. By only extracting the least fixed point, RCA avoids generating too many of them. This is useful when generating a description logic TBox because all concepts are well-defined and necessary.

In search or the greatest fixed point, we defined the notion of self-supported lattices and the functions, P and Q, whose fixed points compute these. They are complementary to F and E. The set of interesting results for RCA is characterised as the intersection of the fixed points of such functions ($fp(E) \cap fp(Q)$). We then discussed the combination of Q^∞ and E^∞ to circumscribe this set.

The definitions and results of this chapter have been restricted to RCA^0 for the sake of clarity. Although, this remains to be proved, they should hold for RCA as a whole. Indeed, all definitions can be applied to families of contexts and lattices, the order between them being the product order induced by the piece-wise conjunction. All operations remain monotone and extensive (or anti-extensive) as soon as the selected scaling operations are. This is enough to preserve the results.

We will now consider methods for extracting these other fixed points.

6. Alternative fixed-point extraction methods

Our initial goal was to define which concept lattices could be considered as the result of RCA on a relational context. These are those elements of $fp(E) \cap fp(Q)$. RCA provides a practical algorithm (based on F or E and FCA^*) to find out the smallest of these: $lfp(E)$. We will first discuss the definition of effective procedures to compute this greatest fixed point: $gfp(Q)$ (§6.1). This procedure is structurally isomorphic to RCA. We will then discuss the strategies to obtain all fixed points for RCA^0 (§6.2).

6.1 Computing the greatest fixed point

The RCA algorithm returns the least fixed point of the induced F function. We first discuss on the opportunity to obtain the greatest fixed point. For that purpose, we first discuss a method commonly used with RCA to obtain a greater least fixed point (§6.1.1) and we consider two approaches: computing the function Q (§6.1.2) or dualising RCA (§6.1.3).

6.1.1 Forcing the greatest fixed point

One way to approach the greatest fixed point with the classical RCA is to biase the algorithm[1]. Indeed, we argued that RCA is based on separating two objects as soon as there is a reason for it and keeping them together otherwise. It is possible to provide a reason to separate objects in RCA by simply adding an extra attribute that is distinct for each object (which may be turned into an FCA context through nominal scaling, see Section 2.2.2).

This approach will generate all singleton classes (one per element of G) and all concepts corresponding to 2^G (because each combination of scaled attributes will generate a different extent). Unfortunately, this is not the greatest fixed point for Q, but the greatest fixed point for E, that can be obtained directly (see Section 5.3.2). Indeed, this approach will separate objects which are indeed not distinguishable from the initial data set.

6.1.2 Implementing Q as the inverse of RCA

Computing the greatest fixed point is now quite easy thanks to the investigation into the Q function (Section 5.4). Indeed, this can be considered as the complete inverse of the E function on which RCA is based:

- It starts with largest context, instead of the smallest one.
- It applies $P = FCA^* + \pi^*$, instead of $F = FCA^* + \sigma^*_\Omega$.
- It stops when no purge occurs.

In summary:

- RCA (F and E) uses an expansive approach: starting with the minimal context K^0 and adding concepts which have support (created by the scaling operation);
- (P and Q) adopts a contracting approach: starting with a maximal lattice 2^G and suppressing those proto-concepts which have no support.

Algorithm

This algorithm can be described as:

1. Initial formal contexts: $\{\langle G_x, M^1_x, I^1_x\rangle\}_{x\in X} \leftarrow \{K^{\langle R, \cup_{z\in X} N(G_z)\rangle}_{+\cup^{\varsigma\in\Omega}_{r\in R|r\subseteq G_x\times G_z} D_{\varsigma,r,N(G_z)}}(\langle G_x, M^0_x, I^0_x\rangle)\}_{x\in X}$.

[1] Marianne Huchard, 2021-07-01.

2. $\{L_x^t\}_{x\in X} \leftarrow FCA^*(\{\langle G_x, M_x^t, I_x^t\rangle\}_{x\in X})$ (or, for each formal context, $\langle G_x, M_x^t, I_x^t\rangle$ the corresponding concept lattice $L_x^t = FCA(\langle G_x, M_x^t, I_x^t\rangle)$ is created using FCA).
3. $\{\langle G_x, M_x^{t+1}, I_x^{t+1}\rangle\}_{x\in X} \leftarrow \pi^*(\{\langle G_x, M_x^t, I_x^t\rangle\}_{x\in X}, R, \{L_x^t\}_{x\in X})$ (i.e. purging is applied, for each relation r_y whose codomain lattice had retracted concepts, generating new contexts $\langle G_x, M_x^{t+1}, I_x^{t+1}\rangle$ including both plain and relational attributes in M_x^{t+1}).
4. If $\exists x \in X; M_x^{t+1} \neq M_x^t$ (purging has occurred), go to Step 2.
5. Return: $\{L_x^t\}_{x\in X}$.

This is exactly the same as the RCA algorithm. The only changes are (*a*) the starting point ($\{K^{\langle R, \cup_{z\in X} N(G_z)\rangle}_{+\cup_{r\in R | r\subseteq G_x\times G_z}^{\varsigma\in\Omega} D_{\varsigma,r,N(G_z)}}$ ($\langle G$ which replaces K^0 at 1), and (*b*) the purging operation (π^*) which replaces scaling (σ_Ω^*) at 3).

Just like $RCA(K^0, R, \Omega) = E^\infty(FCA^*(K))$, this algorithm computes $gfp(Q) = Q^\infty(FCA^*(\langle G, D_{\Omega,R,N(G)}, I^\top\rangle))$. By Proposition 20, this is the greatest element of $fp(E) \cap fp(Q)$

Example

This algorithm may be applied to the example of Section 4.2, starting with an ABox containing four objects:

$$A = \{\top(a), \top(b), \top(c), \top(d), p(a,b), p(b,a), p(c,d), p(d,c), p(a,a), p(b,b)\}$$

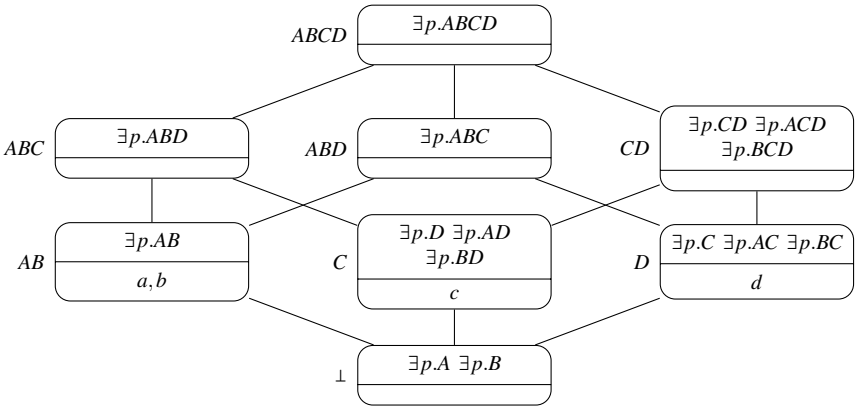
The relation may be encoded as the relation:

p	a	b	c	d
a	×	×		
b	×	×		
c				×
d			×	

Which leads to the following context scaled to $D_{\{p\},\{\exists\},2^G}$:

	$\exists p.ABCD$	$\exists p.ABC$	$\exists p.ABD$	$\exists p.ACD$	$\exists p.BCD$	$\exists p.CD$	$\exists p.BD$	$\exists p.BC$	$\exists p.AD$	$\exists p.AC$	$\exists p.AB$	$\exists p.A$	$\exists p.B$	$\exists p.C$	$\exists p.D$
a	×	×	×								×				
b	×	×	×								×				
c	×		×	×	×	×	×		×						×
d	×	×		×	×	×		×		×				×	

Which when applied FCA provides:



In this lattice, the attributes:
– $\exists p.ACD$ and $\exists p.BCD$ in CD,
– $\exists p.AD$ and $\exists p.BD$ in C,
– $\exists p.AC$ and $\exists p.BC$ in D, and
– $\exists p.A$ and $\exists p.B$ in $\bot$,
are not supported and thus should be purged from the initial context (in Step 3) which becomes:

	$\exists p.ABCD$	$\exists p.ABC$	$\exists p.ABD$	$\exists p.CD$	$\exists p.AB$	$\exists p.C$	$\exists p.D$
a	×	×	×		×		
b	×	×	×		×		
c	×		×	×			×
d	×	×		×		×	

From which FCA (in Step 2) yields the Lattice L_3 of Figure 4.3 (Section 4.2).

Analysis

The complexity of this approach depends on the size of the set of attributes $2^{|G|} \times |R| \times |\Omega|$. Considering that $|R| \times \Omega \ll |G|$, the worst case should occur when the algorithm suppresses a minimal number of elements at each iteration. However, for an iteration to induce another iteration, it is necessary that at least one concept is suppressed. Hence, the number of iterations is bound by the number of concepts, i.e. $2^{|G|}$. Assuming, that the complexity of the operations, σ^* and FCA^* in the loop is not higher, the complexity of this algorithm, for RCA^0 should thus be considered in O($2^{|G|}$).

6.1.3 Dualisation

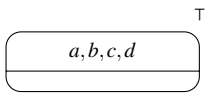
Petko Valtchev[2] suggested to use the dual lattice and added inverting 1 and 0. This would have the advantage to enable the use of the same RCA algorithm for computing both E^∞ and Q^∞ by performing something like: $Q^\infty(L) = (E^\infty(L^D))^D$ in which $\cdot^D$ is a dualisation operator.

In formal concept analysis, dualisation has a precise meaning: objects become attributes, attributes become objects and the incidence relation is inverted (one uses: $\langle M, G, I^{-1} \rangle$). The provided result is an isomorphic lattice (top and bottom inverted if drawn as a standard concept lattice).

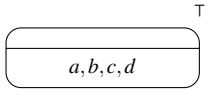
This raises one question: what is the dual of an *RCA* problem? This question is interesting in itself. It is obvious that it should take the dual of *FCA* into account.

This also raises one remark: since it will provide an isomorphic lattice and not a different lattice, this does not seems to be what we are after.

In RCA^0, this would lead to concepts lattices made of:



instead of:



[2]Personal communication, 2019-09-25.

The problem is then that there are no objects to be related by the relational contexts. There is thus a deeper issue which is the definition of the dual of RCA.

We tried below to apply this to the independent problem to use the complement of the relational contexts, but this does not provide any insight.

Algorithm

Hence, the result obtained by dualisation would be:
1. Take the complement relational context (replacing 1 by 0);
2. Perform RCA on it;
3. Take the dual of the result.

Example

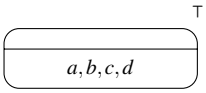
Let see how it works on the example above (Section 4.2). We start with an abox containing four objects: $A = \{\top(a), \top(b), \top(c), \top(d), p(a,b), p(b,a), p(c,d), p(d,c)\}$

The dual is thus: $A = \{\top(a), \top(b), \top(c), \top(d), \bar{p}(a,a), \bar{p}(a,c), \bar{p}(a,d), \bar{p}(b,b), \bar{p}(b,c), \bar{p}(b,d), \bar{p}(c,a), \bar{p}(c,b), \bar{p}(c,c), \bar{p}(d,a), \bar{p}(d,b), \bar{p}(d,d),\}$

Which can be encoded as an empty formal context for $\top$ and the relational self-context:

$R_{\bar{p}}$	a	c	b	d
a	×	×		×
c	×	×	×	
b		×	×	×
d	×		×	×

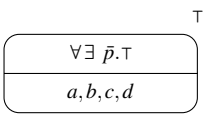
The empty context will generate the singleton lattice:



The application of the scaling operation provides the attribute $\forall\exists\ \bar{p}.\top$ which generates a new context:

	$\forall\exists\ \bar{p}.\top$
a	×
c	×
b	×
d	×

Since this is the same as in Section 4.2, this will return the same result:



At that point, it is not very clear what applying dualisation again means. It is very likely that instead of using $\sigma_{\forall\exists}$ we should have applied a dual scaling operation. However, it is unclear how to define it.

Hence, this approach does not work, at least for RCA^0, but RCA^0 belongs to RCA so this approach (and any variant taking the dual of the formal context does not work either).

However this dualising idea is interesting and should work at some point.

6.2 Extracting all fixed points in RCA^0

Besides the least and greatest fixed point, it is interesting to be able to extract all elements of $fp(E) \cap fp(Q)$. Unfortunately, as shown in Property 20, this is not the interval sublattice $[lfp(E)\ gfp(Q)]$.

We discuss different methods to compute them.

6.2.1 Naïve strategy: the full lattice approach

We end up with two functions, complementary in their structure, one expanding the context, the other contracting it. In the perspective of enumerating all self-supported fixed points, it is tempting to either start from $lfp(E)$ and use E or start from $gfp(Q)$ and use Q. Unfortunately, these starting points being fixed points for these very functions, this leads nowhere. It is necessary to escape the fixed points.

The naïve strategy would simply consist of generating all elements of the interval and testing them for being fixed points. There are several questions to answer in doing so:

- Where to perform the exploration? In $\mathcal{L}$ or $\mathcal{K}$.
- How to perform the exploration? Starting from the least fixed point and adding concepts/attributes or starting from the greatest fixed point and suppressing concepts/attributes.
- How to perform the test?

As illustrated by the example of Section 4.3, the extrema of $\mathcal{L}$ may also be those of $fp(E) \cap fp(Q)$. In consequence, it may be necessary to explore the whole space $\mathcal{L}$ or $\mathcal{K}$. It is more efficient in theory to perform the exploration on lattices because the number of concepts is bounded by $2^{|G|}$, so the number of lattices to test is bounded by $2^{2^{|G|}}$. In the context space, this number is the same in RCA^0, however as soon as more relations or scaling operators are considered, it can be larger.

Performing the exploration from the greatest fixed point or least fixed point does not matter and can be performed in the same manner. The interval can be generated by suppressing from $gfp(Q)$ a subset of $gfp(Q) \setminus lfp(E)$. Alternatively, it can be obtained by adding to $lfp(E)$ the same subsets of $gfp(Q) \setminus lfp(E)$.

Concerning the test, from these lattices of the interval it is necessary to suppress those: (a) which are not concept lattices; (b) which are not self-supported, i.e. fixed points for Q; (c) which are not closed, i.e. fixed points for E. Given L and knowing $\kappa^*(L)$, it is simpler to test:

- if $FCA^*(\kappa^*(L)) = L$, then L is a valid concept lattice, and
- if $\sigma^*_\Omega(L) = \kappa^*(L)$, then $L \in fp(E)$, and
- if $\pi^*(L) = \kappa^*(L)$, then $L \in fp(Q)$.

Algorithm

The naïve algorithm is:

1. Initialisation: $R \leftarrow \varnothing$
2. For each $C \subseteq gfp(Q) \setminus lfp(E)$ do
   (i) $L \leftarrow lfp(E) \cup C$ (or $gfp(Q) \setminus C$)
   (ii) if $\sigma^*_\Omega(L) = \kappa^*(L)$ and $\pi^*(L) = \kappa^*(L)$ and $FCA^*(\kappa^*(L)) = L$, then $R \leftarrow R \cup \{L\}$
3. return R

An example is provided for the next approach as it extends this one in a more interesting way.

6.2.2 A navigation-based approach

The functions π^* and σ^*_Ω used in the test are those of the function E and Q respectively. Hence, it should be worth exploiting the information they provide and not stop here. Indeed:

- if $L \in fp(E) \cap fp(Q)$, then L is a fixed point,

– if $L \in fp(E)$, then apply Q until a fixed point is reached, this is still a fixed point for E (Property 17),
– if $L \in fp(Q)$, then apply E until a fixed point is reached, this is still a fixed point for Q (Property 18),
– otherwise, E and Q can be applied in parallel, one and then the other, this may lead to two different fixed points (Property 26), but actually, the best strategy is to do nothing.

Algorithm

This suggests an algorithm marking each lattice:

1. Initialisation: $R \leftarrow \varnothing$
2. For each $C \subseteq gfp(Q) \setminus lfp(E)$ do
   (*i*) $L \leftarrow lfp(E) \cup C$ (or $gfp(Q) \setminus C$)
   (*ii*) if $\neg marked(L)$ then
   - $mark(L)$
   - if $\sigma^*_\Omega(L) = \kappa^*(L)$, then
     - if $\pi^*(L) = \kappa^*(L)$ and $FCA^*(\kappa^*(L)) = L$, then $R \leftarrow R \cup \{L\}$
     - else
       * $L' \leftarrow Q(L)$
       * while $L' \neq L \wedge \neg$marked(L') do
         · $mark(L')$
         · $L \leftarrow L'$
         · $L' \leftarrow Q(L)$;
       * if $L = L'$ and $FCA^*(\kappa^*(L)) = L$, then $R \leftarrow R \cup \{L\}$
   - elseif $\pi^*(L) = \kappa^*(L)$, then
     - $L' \leftarrow E(L)$
     - while $L' \neq L \wedge \neg$marked(L') do
       * $mark(L')$
       * $L \leftarrow L'$
       * $L' \leftarrow E(L)$;
     - if $L = L'$ and $FCA^*(\kappa^*(L)) = L$ then $R \leftarrow R \cup \{L\}$
3. return R

Example

Here it is applied to the example of Section 4.2. $gfp(Q)$ is L_3 and $lfp(E)$ is L_1, hence $gfp(Q) \setminus lfp(E)$ is $\{\bot, C, D, AB, CD, ABC, ABD\}$. This set contains 128 subsets. Each of these must be added to L_1 and tested for closure. Let consider some of them:

– Let start with $\varnothing$, $lfp(E) \cup \varnothing = L_1$, which can be marked and added to R as it is known to be a fixed point.
– Then consider $\{C\}$, $L = L_1 \cup \{C\}$ is not marked. $\sigma^*_\Omega(L) \neq \kappa^*(L)$ because it would add the attribute $\exists p.C$ and $\pi^*(L) \neq \kappa^*(L)$ because $\exists p.D$ in C is not supported.
– ...
– Consider $L = L_1 \cup \{AB\}$, this time $\sigma^*_\Omega(L) = \kappa^*(L)$ and $\pi^*(L) = \kappa^*(L)$, moreover, it is a valid concept lattice, so it is marked and added to L.
– ...
– Consider $L = L_1 \cup \{C, D\}$, then $\sigma^*_\Omega(L) = \kappa^*(L)$ and $\pi^*(L) = \kappa^*(L)$, but this is not a valid concept lattice, hence it is marked,
– we turn into $L = L_1 \cup \{C, D, C \wedge D\} = L_1 \cup \{C, D, \bot\}$, which is a valid concept lattice satisfying $\sigma^*_\Omega(L) = \kappa^*(L) = \pi^*(L)$, so it is marked and added to R.
– ...

This example only convers a few cases: a self-supported lattice, an invalid concept lattice, a concept lattice which is neither closed nor self supported. Hence, it triggers no navigation. The problem is that concepts are either self-supported, or supported by a concept which depends on itself, hence, as soon as one of them is missing the resulting lattice is neither self-supported nor closed. This remark actually applies to all the examples of Chapter 4.

p	a	b	c	d
a	×	×	×	
b	×	×		×
c				
d				

q	a	b	c	d
a				×
b				×
c				
d				

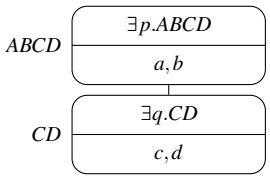


Figure 6.1: a and b are undistinguishable; c and d are recursively undistinguishable: they are not in relation with the same objects, but with undistinguishable objects.

Analysis

The complexity of this approach is given by the size of the initial lattice ($2^{|G|}$). It is the worst case because the lattice cannot grow. This worst case for RCA^0 is actually the worst case for *RCA*^{1} as adding properties does not allow to add more concepts (at least if $|M| \ll |G|$) and the worst case for *RCA* (because $2^n > 2^m + 2^{n-m}$).

Is there a possibility to reduce the initial lattice size? Yes of course, like for FCA it suffices to not consider undistinguishable objects (and non distinguishable properties and relations). In FCA, undistinguishable objects are those corresponding to the same row in the incidence table and undistinguishable properties those which correspond to the same column. For properties, this does not change. Moreover, we may also consider as undistinguishable relations those which have the same relational context. However, what would reduce the size of the lattice are undistinguishable objects. Here, what makes objects undistinguishable (Example 3) is that they have (a) the same property values (if we are not in RCA^0), and (b) that they are related to the exact same objects (or recursively, undistinguishable objects). If they are embedded in an isomorphic structure, they are not undistinguishable because it is sufficient to have distinct classes for each objects for making them distinguishable. Note that

Example 3 (Recursively undistinguishable objects)**.** *Consider an ABox given by* $\{p(a,b), p(a,a), p(b,a), p(b,b), p(c,a), p(d,b)\}$. *a and b are undistinguishable because they have the same column in p's relation. This is not the case of c and d, the former being only in relation with a and the latter with b. However, since a and b are undistinguishable, there will be no concept in the extent of which a is and not b. Hence, it will never be possible to distinguish between c and d either. Worse, c and d are, in this case, non distinguishable from a and b (that could be different, for instance by adding a relation q such that* $q(c,d)$ *and* $q(d,d)$*, then c and d would still be undistinguishable, but distinguishable from a and b, see Figure 6.1).*

Note that the notion of undistinguishability is strongly dependent on Ω.

6.2.3 The closedness condition

In a concept lattice, there is one minimal concept in which each object belong and a maximal concept in which one attribute belong. Any sublattice which does not satisfy these conditions is proscribed.

This means that

- when suppressing the minimal concept for an object, there must remain a single minimal superconcept of it, and
- when suppressing the maximal concept for an attribute, there must remain a single maximal subconcept of it.

Said otherwise, for each minimal concept (resp. maximal concept), there must remain a single most specific subsumer (resp. most general subsumee). This is illustrated by Figure 6.2.

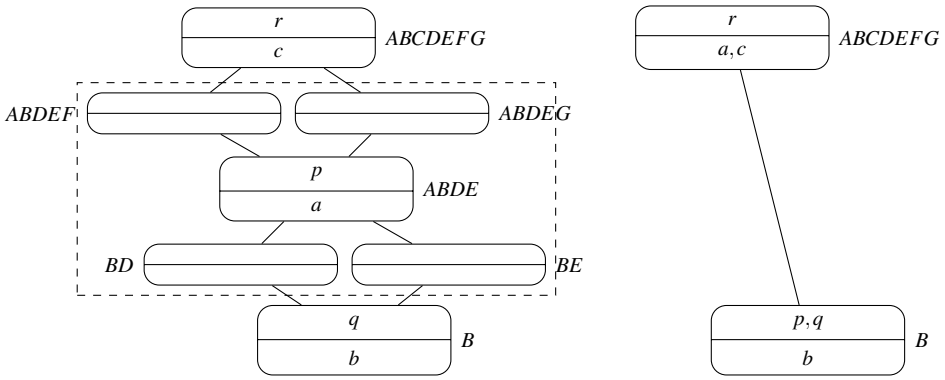


Figure 6.2: What must happen for suppressing a set of concepts: it is not possible to suppress $ABDE$ if BD abd BE remain or if $ABDEF$ and $ABDEG$ remain

Example

In L_3 (Figure 4.3), there are three minimal concepts AB, C and D. It is also possible to treat $\perp$ as minimal because if it does not satisfy this condition, then its suppression does not return a lattice. This indicates that:

R1 it is not possible to suppress AB if ABC and ABD are present;
R2 it is not possible to suppress C if ABC and CD are present;
R3 it is not possible to suppress D if ABD and CD are present;
R4 it is not possible to suppress $\perp$ if AB and C or AB and D or C and D are present.

Keeping track of these dependencies avoids testing violations.

In L_3 all concepts but $\perp$ are maximal. However, when suppressing a set of concept it is only necessary to take care of those attributes which are preserved by the suppression, i.e. those who refer to non suppressed concepts. In particular, those attributes which refer to their maximal concepts do not have to be taken into account. There are only four attributes and as many concepts not in this case in L_3:

– $\exists p.C$ in D
– $\exists p.D$ in C
– $\exists p.ABC$ in ABD
– $\exists p.ABD$ in ABC

The rule above indicates that:

– C and D each having one single subsumee, they could be suppressed without violating this constraint;

R5 it is not possible to suppress ABC if AB and C are present;
R6 it is not possible to suppress ABD if AB and D are present.

This means, for instance, that if one wants to suppress ABD, AB or D and CD must be suppressed.

These constraints have to be generalised because, on larger lattices, a concept suppression may depend on a large variety of constraint suppressions. For instance, it may seem from above that $\perp$ can be suppressed if there does not remain two of AB, C and D. This is not true, actually it is not possible to suppress C, D and $\perp$ only since the result would still not be a lattice. This is hidden by the fact that we also know that C or D cannot be suppressed for other reasons.

As we will see in the next section, there will be other constraints applying.

Algorithm

This can be summarised as a predicate $closed(C)$ applying to a set of contexts:

$$\begin{aligned} min(C,g) &= \mu_{\leq}\{c \in C | o \in extent(c)\} \\ max(C,m) &= \mu_{\geq}\{c \in C | m \in intent(c)\} \\ closed(C) &\text{ iff } \forall g \in G, |min(C,g)| = 1 \wedge \forall m \in M, |max(C,m)| = 1 \end{aligned}$$

Analysis

Such measure does not affect the wost case complexity but can reduce the number of lattices to test. This is summarised on the running example below:

	Theoretical worst case	Example
number of classes in $gfp(Q)$	$2^{\lvert G\rvert} - 1 = 15$	7
number of lattice to check	$2^{2^{\lvert G\rvert}-1} = 32758$	$2^{8-1} = 128$
number of lattices cancelled by minimum (R1–4)		46
number of lattices cancelled by maximum (R5–6)		28
number of lattices cancelled by both		10
number of lattices remaining to check		64

6.2.4 The support graph approach

A smarter strategy consists of analysing the sets of concepts in $gfp(Q)$ that support each others through references, and adding these one by one to $lfp(E)$ or suppressing them from $gfp(Q)$.

Indeed, the previous approaches suppress concepts from $gfp(P)$ or add them to $lfp(F)$ one by one. However, unless self-supported, i.e. refering only to concepts that depend on itself, a concept will not be supported.

It would be more worthy to determine sets of concepts that supports each others and add such sets of concepts. Of course, this is not very simple because some set of concepts may be required to introduce or suppress others.

For that purpose, we introduce a dependency graph between concepts requiring other concepts to be introduced. Precisely, this graph is the graph of the relation $\rightsquigarrow \subseteq C \times C$ defined as:

$$c \rightsquigarrow c' \text{ iff } \exists r; (\exists r.c') \in intent(c)$$

Two concepts which relate to each others through this relation cannot be suppressed or added independently. This can be generalised to the strongly connected components of the graph of $\rightsquigarrow$. This simple observation leads to add or suppress concepts components per components and not concept per concept.

Moreover, the result of adding or suppressing a component from a lattice, should lead to a new lattice. In particular, it must satisfy the closedness condition (Section 6.2.3). The rules considered in Section 6.2.3 have now to be applied to components:

– when suppressing the component of the minimal concept for an object, there must remain a single minimal super concept of it, and
– when suppressing the component of the maximal concept for an attribute, there must remain a single maximal subconcept of it.

This entails that there must remain a single sub/super-component, but in addition that it must not contain more than a single sub/super-concept.

Although these principles work well on the example below, the derived algorithm is unlikely to be correct in all cases. Indeed, in FCA, the attributes in concepts may be present for two reasons: because

SCC	name	intent	sup	sub	depends
0	$ABCD$	$\exists p.ABCD$	*	ABC, ABD, CD	
1	ABC	$\exists p.ABCD$, $\exists p.ABD$	$ABCD$	AB, C	0
	ABD	$\exists p.ABCD$, $\exists p.ABC$	$ABCD$	AB, D	0
2	CD	$\exists p.ABCD$, $\exists p.CD$	$ABCD$	C, D	0
3	AB	$\exists p.ABCD$, $\exists p.ABD$, $\exists p.ABC$, $\exists p.AB$	ABC, ABD	$\bot$	1
4	C	$\exists p.ABCD$, $\exists p.ABD$, $\exists p.CD$, $\exists p.D$	ABC, CD	$\bot$	1,2
	D	$\exists p.ABCD$, $\exists p.ABC$, $\exists p.CD$, $\exists p.C$	ABD, CD	$\bot$	1,2
5	$\bot$	$\exists p.ABCD$, $\exists p.ABD$, $\exists p.ABC$, $\exists p.AB$, $\exists p.CD$, $\exists p.D$, $\exists p.C$	AB, C, D		2,4

Table 6.1: The nodes of L_3 organised in strongly connected components.

they are necessary to the existence of the concept (typically because the concept is a meet of two other concepts), or because they are accidental, i.e. they happen to apply to all objects of the extent, but the concept is justified by the other attributes. In principle, defining the dependency graph based on necessary attributes is sufficient. However, an accidental attribute in one lattice may be necessary (for the same concept) in a different lattice. Hence, the graph and its strongly connected components may change. This calls for further studies to go beyond RCA^0.

Algorithm

The algorithm first computes $gfp(Q)$ and $lfp(E)$ in order to take them as a starting point and to determine which concepts have to be considered. Then it determines the set of strongly connected components of the $\rightsquigarrow$ relation. Finally it enumerates the subsets of this set which can be added to $gfp(G)$ or suppressed from $lfp(E)$.

1. Compute $gfp(Q)$ and $lfp(E)$
2. $C \leftarrow gfp(Q) \setminus lfp(E)$
3. Compute S the set of strongly connected components of the graph of $\rightsquigarrow$ over C.
4. $R \leftarrow \varnothing$
5. For each $S' \subseteq S$ do
   - $L \leftarrow lfp(E) \cup \bigcup S'$
   - if $closed(L)$, then $R \leftarrow R \cup \{L\}$
6. Return R

The same algorithm could be described as suppressing the components from $gfp(Q)$ as used in the example below.

The algorithm uses a test that the union of the strongly connected components satisfies the closedness condition. However, this can be implemented more efficiently (see below).

Example

Take the example of Section 4.2, $gfp(Q) = L_3$ (and $lfp(E) = L_1$). The concepts of L_3 are enumerated in Table 6.1.

In principle, there are $2^5 = 32$ possible suppressions (the concepts of $lfp(E)$, here $ABCD$ cannot be suppressed). Note that, among these 32 items (Table 6.2), the lattice $L^{\#}$ displayed in Figure 5.6, which belongs to the interval, is not even considered.

The minimal and maximal concept constraints identified in the previous section, can now be applied to the components to which the concepts belong. Looking at the dependency graph of Figure 6.3, it happens that:

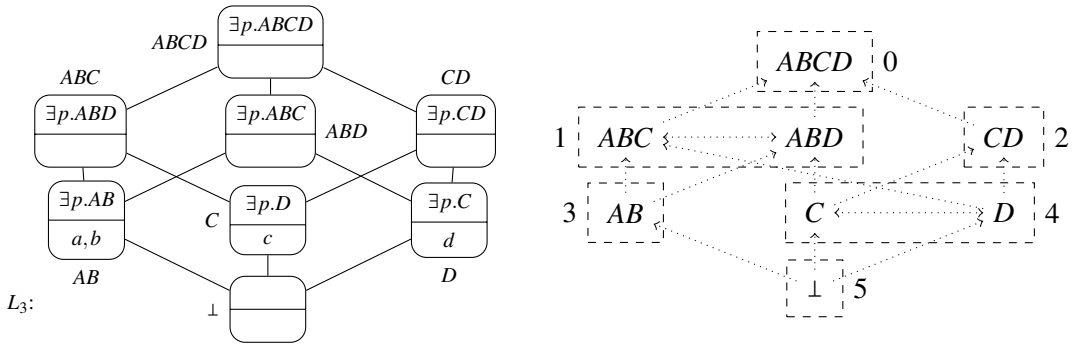


Figure 6.3: The strongly connected components of the dependency graph of $gfp(Q) = L_3$.

r1 3 cannot be suppressed if 1 is preserved (previous R1);
r2 5 cannot be preserved if 4 is preserved (part of previous R3);
r3 5 cannot be preserved if 2 and 3 are preserved (extension of previous R3);
r4 4 cannot be preserved if 1 and 2 are preserved (merge of R2 and R3).
The maximal rules, concerning attributes, must be adapted. As before, when suppressing a set of concepts, it is only necessary to take care of those attributes which are preserved by the suppression, i.e. those who refer to non suppressed concepts. Here since concepts *ABC* and *ADB* belong to the same component, they are always suppressed together. Hence, rules R5 and R6 are not necessary. This is certainly due to the same reason as observed before: when a concept refers to another, this one refers back to the initial one in these examples.

These rules can be checked on the 32 configurations of Table 6.2.

The 15 remaining fixed points are presented in Figure 6.4. This illustrates Property 23 that it is a complete sublattice of $[lfp(E), gfp(Q)]$.

Analysis

In principle, the number of classes in $gfp(Q)$ is in $O(2^{|G|})$. Hence the number of lattices to test for closedness is in $O(2^{2^{|G|}})$. The number of strongly connected components of the dependency graph is still in $O(2^{|G|})$ (for instance with the diagonal relation matrix). Hence, the number of lattice to consider is still in $O(2^{2^{|G|}})$.

In practice, on the given example, designed far before the algorithms, $L_3 = gfp(Q)$ contains 8 classes and $L_1 = lfp(E)$ contains only one, hence there are 128 lattices in the interval $[lfp(E), gfp(Q)]$. Strongly connected components leads to deal with only 5 components hence $2^5 = 32$ alternatives. In fact, using the depdendency techniques, we directly have the 15 alternatives.

This is summarised below:

	Theoretical worst case	Example
number of classes in $gfp(Q)$	$2^{\|G\|} - 1 = 15$	7
number of lattice to check	$2^{2^{\|G\|}-1} = 32758$	$2^{8-1} = 128$
number of strongly connected components	$2^{\|G\|} = 16$	6
number of lattices to check	$2^{2^{\|G\|}} = 65536$	$2^{6-1} = 32$
number of lattices cancelled by minimum (r1–r4)		17
number of lattices cancelled by maximum		0
number of fixed points		15

Name	Suppressed SCC	r1	r2	r3	r4	Added SCC
L_3	∅				12345	
	1					2345
L_4	2					1345
	3	×				1245
	4				×	1235
	5		×	×		1234
$Q(L^{\#})$	12					12
	13					13
L_2	14					235
	15		×	×		234
	23	×				145
	24		×	×		135
	25					134
	34	×			×	125
	35	×		×		124
	45			×	×	123
	123					45
	124					35
	125		×	×		34
	134					25
	135		×	×		24
	145			×	×	23
	234	×				15
	235	×	×			14
	245					245
	345	×		×	×	13
	1234					5
	1235		×			4
	1245					3
	1345					2
	2345	×				1
L_1	12345					∅

Table 6.2: All possible subsets of strongly connected components and the constraints that they eventually violate.

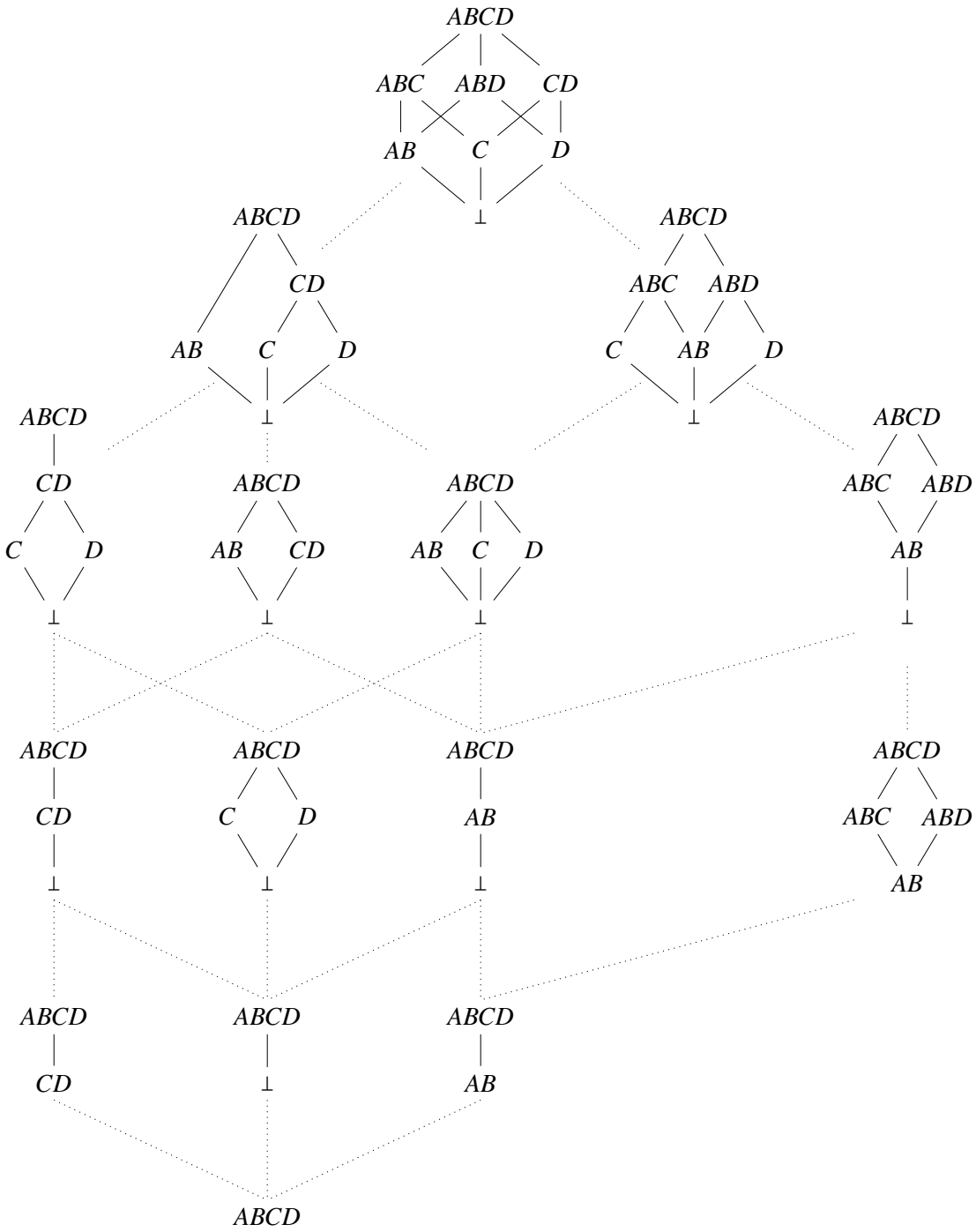


Figure 6.4: All the lattices belonging to $fp(E) \cap fp(Q)$ in the example of Section 4.2.

Conclusion

We provided effective procedures to compute $gfp(Q)$ and $fp(E) \cap fp(Q)$ in RCA^0. The former should easily be extended to RCA contexts as it is very closed to RCA. It will be more difficult to apply the latter to RCA due to dependencies across lattices. It should first be tested with several scaling operators and several properties, before adding numerous contexts.

Concerning the extraction of all fixed points, dependencies between different contexts will have to be taken into account.

Moreover, if the goal is to extract exactly one link key from each concept lattices, this can be used to reduce the number of strongly connected components. Indeed, all components containing more than one concepts have to be discarded since they will not provide a link key to be retained in the family.

7. Conclusion

We discussed the difficult issue of circular dependencies when extracting link keys. We showed (Chapter 3) that the results of Deliverable 1.1 for link key extraction with RCA can be extended to extract families of link keys in the presence of circular dependencies.

It is also an illustration of the general-purpose aspect of RCA techniques, usually achieved by introducing new scaling operators [Braud et al. 2018]. It is exploited here to extract link key candidates instead of concept descriptions.

However, Chapter 4 raised issues about the nature of the results provided in this case. The problem concerning link key extraction is that not all valid link keys may be returned. More generally, the problem is that RCA does not necessarily returns all valid (closed) concepts. This motivated a reformulation of the RCA semantics in terms of fixed points of the functions at the core of RCA.

We studied this problem in a minimalistic version of RCA called RCA^0. In Chapter 5, we gave a new, fixed-point based, semantics for RCA^0 and proved that the traditional RCA algorithm computes the least fixed point. However, the extraction of other fixed points may be interesting, so we discussed alternative RCA algorithms, first returning the greatest fixed point (useful for link key extraction). We also identified as self-supported fixed points those other fixed points of interest. The least fixed point being the smaller of these. This led to develop another function which allows extracting the greatest of them as an alternative to RCA.

In Chapter 6, we also presented an algorithm, inverse of RCA, to extract the greatest fixed point. We discussed techniques by which it is possible to enumerate all relevant fixed points.

Applying these results to link key extraction, the least fixed point contains all grounded links; the greatest fixed point contains self-supported links (or rather recursively-supported links). The approach taken by RCA is a cautious/skeptical approach accepting only links within the least fixed-point semantics. A more risky/credulous position would adopt those links which are in the greatest fixed point. As shown by the example of Section 4.4, the most promising link key candidates may be found within the greatest fixed point.

This is not something specific to link key extraction, RCA users often have to provide extra attributes in order to increase discrimination and thus biasing the RCA results to a greater least fixed point (see Section 6.1.1). An interesting remark is that if the identification of objects has to be given through relations and not a simple set of attributes, then this is a key and this key may be induced by the method provided in Chapter 3. Then a fixed point has to be sought between the key identifying objects, used to generate concepts, and concepts, used to generate key candidates. This interaction between key and concept extraction may be considered in the context of the logical interpretation of relational scaling (Section 2.2.5) which provides a way to integrate keys as extra knowledge used in concept and key extraction.

There are two remaining issues more directly related to this work to address next:

– Extending these results to RCA as a whole,
– Applying them to link key extraction.

We are confident that the theoretical results provided here can be directly extended to RCA. The algorithms will require some more care as they will necessitate to investigate the dependencies between different contexts and lattices.

8. Bibliography

Al-Bakri, Mustafa, Manuel Atencia, Steffen Lalande, and Marie-Christine Rousset (2015). "Inferring same-as facts from Linked Data: an iterative import-by-query approach". In: *Proc. 29th AAAI Conference on Artificial Intelligence, Austin (TX US)*. AAAI Press, pp. 9–15 (cit. on p. 5).

Atencia, Manuel, Jérôme David, and Jérôme Euzenat (2014). "Data interlinking through robust linkkey extraction". In: *Proc. 21st European Conference on Artificial Intelligence (ECAI)*. IOS Press, pp. 15–20 (cit. on pp. 5, 6, 24).

Atencia, Manuel, Jérôme David, and Jérôme Euzenat (2014). "What can FCA do for database linkkey extraction?" en. In: *Proc. 3rd ECAI workshop on What can FCA do for Artificial Intelligence? (FCA4AI), Praha (CZ)*. Vol. 1257. CEUR Workshop Proceedings. CEUR-WS.org, pp. 85–92 (cit. on p. 5).

— (2021). "On the relation between keys and link keys for data interlinking". In: *Semantic web journal*. to appear.

Atencia, Manuel, Jérôme David, Jérôme Euzenat, Amedeo Napoli, and Jérémy Vizzini (2020). "Link key candidate extraction with relational concept analysis". In: *Discrete applied mathematics* 273, pp. 2–20 (cit. on pp. 2, 5, 6, 9, 15).

Baader, Franz, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, eds. (2003). *The description logic handbook: theory, implementations and applications*. Cambridge University Press (cit. on pp. 9, 11).

Belohlávek, Radim (2008). *Introduction to formal concept analysis*. Tech. rep. Univerzita Palackého, Olomouc (CZ) (cit. on p. 25).

Bizer, Chris, Tom Heath, and Tim Berners-Lee (2009). "Linked data — the story so far". In: *International Journal of Semantic Web Information Systems* 5.3, pp. 1–22 (cit. on p. 5).

Braud, Agnès, Xavier Dolques, Marianne Huchard, and Florence Le Ber (2018). "Generalization effect of quantifiers in a classification based on relational concept analysis". In: *Knowledge-based systems* 160, pp. 119–135 (cit. on pp. 9, 10, 55).

Euzenat, Jérôme (2021). "Fixed-point semantics for barebone relational concept analysis". In: *Proc. 16th international conference on formal concept analysis (ICFCA), Strasbourg (FR)*. Vol. 12773. Lecture notes in computer science, pp. 20–37 (cit. on p. 2).

Euzenat, Jérôme and Pavel Shvaiko (2013). *Ontology matching*. en. 2nd. Heidelberg (DE): Springer. 520 pp. (cit. on p. 5).

Euzenat, Jérôme, Manuel Atencia, Jérôme David, Amedeo Napoli, and Jérémy Vizzini (2019). *Candidate link key extraction with formal concept analysis*. Deliverable 1.1. Elker project (cit. on p. 15).

Ferrara, Alfio, Andriy Nikolov, and François Scharffe (2011). "Data Linking for the Semantic Web". In: *International Journal of Semantic Web and Information Systems* 7.3, pp. 46–76 (cit. on p. 5).

Ferré, Sébastien and Peggy Cellier (2020). "Graph-FCA: an extension of formal concept analysis to knowledge graphs". In: *Discrete applied mathematics* 273, pp. 81–102 (cit. on p. 11).

Ganter, Bernhard and Sergei O. Kuznetsov (2001). "Pattern Structures and Their Projections". In: *International Conference on Conceptual Structures (ICCS)*. Ed. by Harry S. Delugach and Gerd Stumme. Vol. 2120. Lecture Notes in Computer Science. Springer, pp. 129–142 (cit. on p. 11).

Ganter, Bernhard and Rudolf Wille (1999). *Formal Concept Analysis: mathematical foundations*. Berlin: Springer (cit. on pp. 5, 7–9, 27).

Gmati, Maroua, Manuel Atencia, and Jérôme Euzenat (2016). "Tableau extensions for reasoning with link keys". In: *Proc. 11th Ontology matchijng workshop (OM), Kobe (JP)*. Vol. 1766. CEUR Workshop Proceedings. CEUR-WS.org, pp. 37–48.

Heath, Tom and Christian Bizer (2011). *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool (cit. on p. 5).

Hogan, Aidan, Antoine Zimmermann, Jürgen Umbrich, Axel Polleres, and Stefan Decker (2012). "Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora". In: *Journal of Web Semantics* 10, pp. 76–110 (cit. on p. 5).

Kaytoue, Mehdi, Sergei O. Kuznetsov, Amedeo Napoli, and Sébastien Duplessis (2011). "Mining Gene Expression Data with Pattern Structures in Formal Concept Analysis". In: *Information Science* 181.10, pp. 1989–2001 (cit. on p. 11).

Keip, Priscilla, Sébastien Ferré, Alain Gutierrez, Marianne Huchard, Pierre Silvie, and Pierre Martin (2020). "Practical Comparison of FCA Extensions to Model Indeterminate Value of Ternary Data". In: *Proc. 15th International Conference on Concept Lattices and Their Applications (CLA), Tallinn (EE)*. Vol. 2668. CEUR Workshop Proceedings, pp. 197–208 (cit. on p. 11).

Kuznetsov, Sergei (2009). "Pattern Structures for Analyzing Complex Data". In: *Proc. International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing (RSFDGrC)*. Vol. 5908. Lecture notes in computer science, pp. 33–44 (cit. on p. 11).

Kötters, Jens (2013). "Concept Lattices of a Relational Structure". In: *Proc. 21th International Conference on Conceptual Structures (ICCS)*. Vol. 7735. Lecture Notes in Computer Science, pp. 301–310 (cit. on p. 11).

Nebel, Bernhard (1990). *Reasoning and revision in hybrid representation systems*. Lecture Notes in Artificial Intelligence 422. Berlin (DE): Springer Verlag (cit. on p. 25).

Nentwig, Markus, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm (2017). "A survey of current Link Discovery frameworks". In: *Semantic Web* 8.3, pp. 419–436. DOI: 10.3233/SW-150210 (cit. on p. 5).

Ngonga Ngomo, Axel-Cyrille and Sören Auer (2011). "LIMES: A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data". In: *Proc. 22nd International Joint Conference on Artificial Intelligence (IJCAI), Barcelona (ES)*. Barcelona (ES), pp. 2312–2317 (cit. on p. 5).

Prediger, Susanne (1997). "Logical Scaling in Formal Concept Analysis". In: *Proc. 5th International Conference on Conceptual Structures (ICCS), Seattle (WA US)*. Vol. 1257. Lecture Notes in Computer Science, pp. 332–341 (cit. on p. 9).

Rouane-Hacene, Mohamed, Marianne Huchard, Amedeo Napoli, and Petko Valtchev (2013a). "Relational Concept Analysis: mining concept lattices from multi-relational data". In: *Annals of Mathematics and Artificial Intelligence* 67.1, pp. 81–108 (cit. on pp. 5, 9–12, 15, 16, 30, 32, 34).

— (2013b). "Soundness and Completeness of Relational Concept Analysis". In: *Proc. 11h International Conference on Formal Concept Analysis (ICFCA)*. Ed. by Peggy Cellier, Felix Distel, and Bernhard Ganter. Vol. 7880. Lecture Notes in Artificial Intelligence. Springer, pp. 228–243 (cit. on pp. 6, 12, 13, 26, 29).

Saïs, Fatiha, Nathalie Pernelle, and Marie-Christine Rousset (2007). "L2R: A Logical Method for Reference Reconciliation". In: *Proc. 22nd National Conference on Artificial Intelligence (AAAI), Vancouver (CA)*. AAAI Press, pp. 329–334 (cit. on p. 5).

Tarski, Alfred (1955). "A lattice-theoretical fixpoint theorem and its applications". In: *Pacific journal of mathematics* 5.2, pp. 285–309 (cit. on p. 27).

Volz, Julius, Christian Bizer, Martin Gaedke, and Georgi Kobilarov (2009). "Silk – A Link Discovery Framework for the Web of Data". In: *Proc. WWW Workshop on Linked Data on the Web, LDOW, Madrid (SP)*. Vol. 538. CEUR Workshop Proceedings. CEUR-WS.org (cit. on p. 5).

Wajnberg, Mickael (2020). "Analyse relationnelle de concepts: une méthode polyvalente pour l'extraction de connaissance". PhD thesis. Université du Québec à Montréal ; Université de Lorraine (cit. on p. 9).