Université Clermont Auvergne

École Doctorale Sciences pour l'Ingénieur de Clermont-Ferrand

Thèse présentée par

Giacomo Kahn

pour obtenir le grade de

Docteur d'Université

Spécialité Informatique

A Structural Study of Lattices, d-Lattices and some Applications in Data Analysis

---

Soutenue publiquement le 12/12/2018 devant le jury constitué de :

M. BEAUDOU Laurent	Maître de conférences, Université Clermont Auvergne	Examinateur
Mme BERTET Karell	Maître de conférences, Université de La Rochelle	Examinatrice
Mme HUCHARD Marianne	Professeur des Universités, Université de Montpellier	Présidente du jury
M. KUZNETSOV Sergei	Professeur, Higher School of Economics Moscow	Rapporteur
M. NAPOLI Amedeo	Directeur de Recherche, CNRS	Rapporteur
M. NOURINE Lhouari	Professeur des Universités, Université Clermont Auvergne	Examinateur
M. RAYNAUD Olivier	Maître de conférences, Université Clermont Auvergne	Directeur de thèse

Remerciements

La page de remerciements est un passage obligé de la rédaction d'une thèse. Les remerciements énoncés ici n'en sont pas moins sincères. Je souhaiterais tout d'abord remercier Olivier Raynaud, mon directeur de thèse, pour son soutien et ses conseils lors des trois ans passés avec lui. Nos discussions, qu'elles soient scientifiques ou informelles ont été d'une grande importance dans la définition de la direction qu'a pu prendre mon travail de recherche.

Je voudrais bien entendu remercier Sergei Kuznetsov et Amedeo Napoli pour le temps consacré à la relecture de mon manuscrit et leurs remarques qui m'ont permis de l'améliorer grandement. Le fait que deux piliers de la communauté treillis/fouille de données relisent mon travail avec autant d'attention m'honore. Leur déplacement depuis Moscou et Nancy également.

Respectivement de La Rochelle et Montpellier, je remercie grandement Karell Bertet et Marianne Huchard d'avoir accepté de venir m'écouter le jour de la soutenance, en tant qu'examinatrice et directrice du jury. La pertinence de leurs questions et les perspectives qu'elles ont ouvertes me laissent penser que je ne devrais pas m'ennuyer dans les temps à venir.

Maintenant les locaux. Je remercie Lhouari Nourine, non seulement pour sa présence en tant qu'examinateur lors de la soutenance, mais aussi depuis mon master, pour m'avoir montré que sa phrase favorite "les treillis sont partout, tout le monde en fait sans le savoir", est effectivement presque vraie. Je remercie également Laurent Beaudou, pour sa présence en tant qu'examinateur, et en tant que co-encadrant non officiel depuis trois ans.

Au cours de mes trois années de thèse, j'ai eu l'occasion de discuter avec de nombreux chercheurs, et j'ai pu tisser des liens de collaboration avec certains d'entre eux. Il serait injuste de ne pas remercier Alexandre Bazin, pour les moments de travail mais pas que, que l'on a partagé depuis le tout premier jour. L'alignement de nos intérêts de recherche, et ses quelques années d'expérience en ont fait un allié de poids. Sa pédagogie abrupte et sa patience déguisée en ont fait un excellent guide. Merci aussi à Jessie, d'avoir noté dès les début que les treillis, c'est cool. Je remercie aussi tous les autres collaborateurs que j'ai pu avoir, proches ou lointains.

Je remercie tout le personnel administratif du LIMOS, en particulier Béa et Séverine qui ont toujours été là pour m'aider, que ce soit rattraper des bourdes administratives ou préparer des pots. Tous les chercheurs du LIMOS, anciens et nouveaux arrivants, ont aussi leur place dans ces remerciements. En particulier, par ordre d'arrivée, Grilo (sans qui je n'aurais même pas fait d'informatique), Vincent, David, Alexis, Antoine, Matthieu, Matthieu, Oscar, Aurélie et Lucas.

En dehors du labo, il y a le monde. Je remercie tous les Sambagogistes pour leur soutien depuis des années et des années, vous êtes des vrais potes. Les répétitions hebdomadaires ont été pour moi un véritable havre de paix, malgré les décibels.

Merci aussi à ma famille, les parents, les nombreux frères et sœurs qui m'ont toujours soutenu et aidé, jusqu'au jour même de la soutenance.

Et pour Laís, **obrigado**.

Contents

List of Figures

List of Tables

Introduction

I start this dissertation in a not-so-formal way, to introduce, and at the same time summarize the content of the next 128 pages. A careful examination of the title page may have given some of it away: we will talk about lattices, d-lattices (we will get to what they are soon) and their applications.

Lattices and d-lattices are mathematical structures that naturally arise when one considers patterns in data. This is one of the main reasons behind their more recent studies: the set of closed patterns from a binary dataset, which is particularly interesting in data analysis, forms a lattice when ordered by set inclusion. If the dataset is multidimensional, meaning that we deal with three or more dimensions, the corresponding structure is a d-lattice, with d the dimension. This correspondence with data analysis alone is enough of a motivation to study lattices and d-lattices. However, we will see over the course of this dissertation that they can be entertaining and elegant structures too.

If, on your way from the title page to the introduction, you ventured into the table of contents, you may have noticed that this thesis is divided into three parts. The first part, called Combinatorics, gathers some results on the cardinality of d-lattices in the extremal case, and in the average case. The results are frightening: they can be enormous. Having discussed the size of those structures, we enter the second part of the dissertation, called Algorithms. There, we consider several approaches to bypass the size of d-lattices. In the Applications chapters, we finally use lattices rather than studying them, and try –once more– to generate only part of the lattices, in order to reduce the number of objects that we need to handle.

What is less apparent is that this thesis stems from an industrial chair at the Clermont Auvergne university. This working relationship between industries and university is what allowed me to conduct this work on theoretical aspects of data analysis, while maintaining a close interest on applications. The path that we will take from Combinatorics to Applications is a good illustration of that relationship.

The diversity of the matters that are discussed in this dissertation is also the outcome of the many encounters that punctuated my three years as a Ph.D. student. All those encounters and opportunities to work on connected themes were made possible only because my advisor, Olivier Raynaud, encouraged me to do so during my time as his student. I can definitely say that my experience of what research can be –a collaboration between complementary brains– has been enriching, inspiring and exciting.

We now present more in detail the content of each chapter.

First chapter The first chapter gives the mathematical preliminaries that reduce the scope from discrete math to order theory, to lattice theory, and finally to formal concept analysis. More specific definitions are given throughout the dissertation, in order to keep this first chapter as short as possible.

Second chapter In the second chapter, we try to answer the question "How big can a d-lattice get?". We give an (almost) exact answer for 3-lattices, and some reflections for the general d case. This chapter presents joint works with Alexandre Bazin, Laurent Beaudou and Khaveh Khorshkhah.

Third chapter In the third chapter, we are not concerned with the average case for the cardinality of d-lattices, rather than the extremal case. We study the average size of lattices and d-lattices under two models. We also study the average size of some implicational bases. This work was triggered by Julien David's visit at the LIMOS, and was conducted partly with Alexandre Bazin (for the average number of implications).

Fourth chapter The fourth chapter, first of the Algorithms part, presents an algorithm that allow to compute the elements of a d-lattice following a change in the data that represents it. It allows to adapt to a change in the dataset (when a new entry is added) without having to compute the whole new d-lattice from scratch. This work finds its roots in the questions that were asked in the second chapter. We developed it in collaboration with Alexandre Bazin and Olivier Raynaud.

Fifth chapter The fifth chapter carries the notions of reduction and introducer concepts to the multidimensional case. This falls within the idea of trying to reduce the curse of dimensionality that comes with d-lattices. This chapter stems from some discussions with Alexandre Bazin.

Sixth chapter The sixth chapter marks the start of a more applied part. Our objective is not to try to reduce the complexity of some applications of lattices but to actually do it. We present two approaches of conceptual navigation in different structures: the first explores introducer posets while the second navigates into a relational lattice family. The question of the navigation was asked to me by two of the LIRMM's brightest stars, Jessie Carbonnel and Marianne Huchard, in cooperation with Alexandre Bazin.

Seventh chapter Finally, the seventh and last chapter of the dissertation uses lattices as a means of classification, in order to implicitly authenticate users in a system. We present a classifier and some experiments on our case study: "are web-browsing logs enough to recognize a user in a system?". Spoiler alert, they are more than enough. This works was conducted with Diyé Dia, Fabien Labernia, Yannick Loiseau and Olivier Raynaud.

Chapter 1

Preliminaries

This chapter is devoted to the presentation of the mathematical objects manipulated in this dissertation. In the next sections, the reader will find definitions and notations related to sets, orders and more particularly lattices, in general and under the framework of Formal Concept Analysis. We also introduce the generalisation of Formal Concept Analysis to the d-dimensional case, called Polyadic Concept Analysis.

1.1 Ordered Sets

We will start our definitions with a class of mathematical objects a bit more general. It is noteworthy to mention that all the sets we manipulate in this dissertation are finite.

What we call a partially ordered set, or *poset* for short, is a set P, together with an order relation $\leq$, that is reflexive, antisymmetric and transitive. It is *partially* ordered because a pair of elements a and b of P can be either comparable, and then $a \leq b$ or $b \leq a$, or incomparable.

A good example is a genealogy tree. In a genealogy tree, two people are either ancestors of one another, or are incomparable. We can (and will) say that everybody can be considered as his own ancestor (for the sake of our example, this illustrates the reflexivity). Someone cannot be ancestor to one of his ancestors, except himself (antisymmetry). Your ancestors don't stop at rank one: an ancestor of one of your ancestors is also yours (transitivity).

Another example is the set of all positive integers $\mathbb{N}$ and the relation $\leq$ (less-or-equal). In this case, no two elements are incomparable. We say that a poset where no elements are incomparable forms a *chain*.

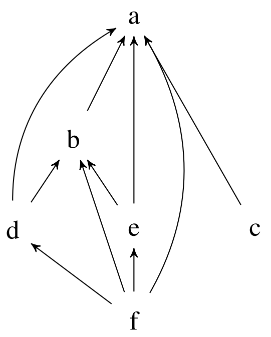


Figure 1.1: A representation of P as a directed acyclic graph, showing all order relations between elements.

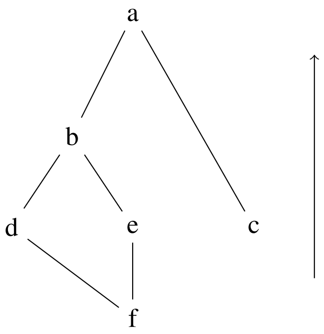


Figure 1.2: Hasse diagram of P, where only the transitive reduction has been kept.

Here, we deal neither with genealogy nor with natural numbers. In most of the cases (if not all), the order relation that we consider is the inclusion relation $\subseteq$ on subsets. As an example, $\{a, b, c\} \subseteq \{a, b, c, d\}$, but $\{a, b, c\}$ and $\{a, b, d\}$ are incomparable, because neither of them is included in the other.

The representation of posets is commonly done using a *Hasse diagram* [1]. A Hasse diagram allows one to embed an order in the plane by representing the elements of P as vertices and the relations as arcs between them. In Figure 1.1, we show the representation of all relations between elements of P. Figure 1.2 shows the Hasse diagram for poset P. In this case, an edge exists between two elements a and b if $a \leq b$ and there is no element c such that $a \leq c$ and $c \leq b$.

In Figure 1.2, an upward arrow shows the direction of the order relation. We omit this arrow when no confusion is likely to occur, and consider that all orders follow this direction: the smallest element is on the bottom, and the greatest at the top, following common sense.

1.2 Lattices

Lattices are a particular class of posets: they have all the properties of partial orders, and some more.

Let $(P, \leq)$ be a poset and S a subset of P. An element x of P, such that for all y in S, $y \leq x$ is called an upper bound of S. A subset of elements can have many upper bounds. An element x of P, such that for all y in S, $y \leq x$ and there is no z such that z is an upper bound of S and $z \leq x$ is called a least upper bound of S. Reciprocally, we can define a lower bound of a subset S of L, and a greatest lower bound.

A *lattice* is a partially ordered set where every pair of elements has a unique greatest lower bound (called *meet*) and a unique least upper bound (called *join*). Additionally, when dealing with finite sets and finite lattices, a lattice has a unique least element (that we call bottom, denoted $\perp$) and a unique greatest element (top, $\top$).

[1] Named after Helmut Hasse, a German mathematician of the 20th century.

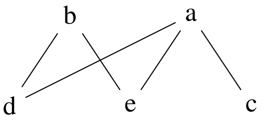


Figure 1.3: An order that is not a lattice

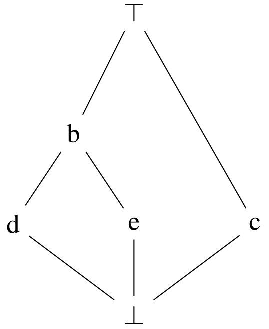


Figure 1.4: A lattice.

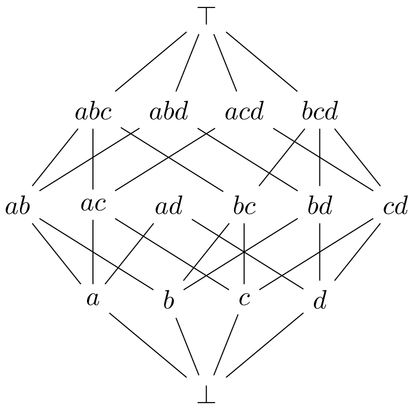


Figure 1.5: A hypercube on 4 elements.

In Fig 1.3, b is an upper bound for elements d and e. Additionally, there is no element “between” d and b, or e and b, so b is a least upper bound for d and e. We can see that a is also a least upper bound for d and e. This order is not a lattice.

In Figure 1.4, one can check that every pair of elements has a unique least upper bound. This is also true for the greatest lower bound. This order is a lattice.

Many classes of lattices are of great interest, however, one that shall be central for this dissertation is the Boolean lattice. A Boolean lattice on a ground set of size n is the lattice of all subsets of $\{1, \ldots, n\}$, ordered by inclusion. Such a lattice is also called a hypercube of dimension n [2] or a powerset of the universe $E = \{1, \ldots, n\}$, and possibly denoted 2^E. Let X and Y be subsets of $\{1, \ldots, n\}$, then the lowest upper bound is given by $X \cup Y$ and the greatest lower bound by $X \cap Y$. A Boolean lattice has 2^n elements. Figure 1.5 shows a Boolean lattice on four elements.

[2] This denomination will not be used much in this dissertation, since we deal with multidimensional datasets, and that a hypercube, be it on a ground set of ten thousand elements, is still of dimension two for us.

1.3 Formal Concept Analysis

Formal Concept Analysis (FCA) is a mathematical framework that revolves around *contexts* as a condensed representation of lattices. This framework allows one to inject the powerful mathematical machinery of lattice theory into data analysis. FCA has been introduced in the 1980's by a research team led by Rudolph Wille in Darmstadt. It is based on previous work by Garrett Birkhoff on lattice theory [1] and by Marc Barbut and Bernard Monjardet [2].

In this section, we give the basic definitions of FCA. For an informative book, the reader can refer to [3].

Definition 1. *A* (formal) context *is a triple* $(\mathcal{O}, \mathcal{A}, \mathcal{R})$ *where* $\mathcal{O}$ *and* $\mathcal{A}$ *are finite sets and* $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$ *is a relation between them. We call* $\mathcal{O}$ *the set of (formal) objects and* $\mathcal{A}$ *the set of (formal) attributes.*

A formal context naturally has a representation as a crosstable, as shown in Figure 1.6. A pair $(o, a) \in \mathcal{R}$ correspond to a cross in cell (o, a) of the crosstable. Such a pair is read "object o has attribute a". Since many datasets can be represented as binary relation such as the one in Figure 1.6, FCA finds natural applications in data analysis.

	a_1	a_2	a_3	a_4	a_5
o_1	×		×		×
o_2		×		×	×
o_3	×	×	×		
o_4			×	×	
o_5	×	×			×
o_6	×		×	×	
o_7	×	×		×	×

Figure 1.6: An example context with $\mathcal{O} = \{o_1, o_2, o_3, o_4, o_5, o_6, o_7\}$ and $\mathcal{A} = \{a_1, a_2, a_3, a_4, a_5\}$. A cross in a cell (o, a) is read "object a has attribute a". A maximal rectangle of crosses is highlighted.

To allow one to efficiently jump from a set of object to the set of attributes that describes it, and vice versa, two derivation operators are defined. For a set O of objects and a set A of attributes, they are defined as follows:

$$\cdot' : 2^{\mathcal{O}} \mapsto 2^{\mathcal{A}}$$

$$O' = \{a \in \mathcal{A} | \forall o \in O, (o, a) \in \mathcal{R}\}$$

and

$$\cdot' : 2^{\mathcal{A}} \mapsto 2^{\mathcal{O}}$$

$$A' = \{o \in \mathcal{O} | \forall a \in A, (o, a) \in \mathcal{R}\}$$

The $\cdot'$ derivation operator maps from a set of objects (resp. attributes) to the set of attributes (resp. objects) that they share. The composition of the two derivation operators forms a Galois connection, and, as such, a closure operator. Depending on which set the composition of operators is applied, we can have two closure operators: $\cdot'' : 2^{\mathcal{O}} \mapsto 2^{\mathcal{O}}$ or $\cdot'' : 2^{\mathcal{A}} \mapsto 2^{\mathcal{A}}$.

Let $O \subseteq \mathcal{O}$ be a set of objects. A closure operator $\cdot''$ is a function from $2^{\mathcal{O}}$ to $2^{\mathcal{O}}$, such that $O \subseteq O''$ (that is, $\cdot''$ is extensive), if $X \subseteq O$ then $X'' \subseteq O''$ ($\cdot''$ is increasing) and $(O'')'' = O''$ ($\cdot''$ is idempotent). The same can be said for the closure operator on sets of attributes. A set X such that $X = X''$ is said to be closed.

Definition 2. *A pair (O, A) where $O \subseteq \mathcal{O}$ and $A \subseteq \mathcal{A}$ are closed, and $A = O'$ and $O = A'$ is called a* concept*. O is called the* extent *of the concept while A is called the* intent *of the concept.*

A concept can be seen as a maximal rectangle of crosses in the crosstable that represent a context, up to permutation on rows and columns. This point of view, while not exactly formal, gives good insight and will be used thoroughly in this dissertation. In Figure 1.6, the concept $(o_2o_7, a_2a_4a_5)$ is highlighted. We denote by $\mathcal{T}(\mathcal{C})$ the set of all concepts of a context $\mathcal{C}$.

We can order the concepts of $\mathcal{T}(\mathcal{C})$. Let (O_1, A_1) and (O_2, A_2) be concepts of a context. We say that (O_1, A_1) is a subconcept of (O_2, A_2) (denoted $(O_1, A_1) \leq (O_2, A_2)$) if $O_1 \subseteq O_2$. As the Galois connection that rise from the derivation is antitone, this is equivalent to $A_2 \subseteq A_1$. The concept (O_2, A_2) is a superconcept of (O_1, A_1) if $O_2 \supseteq O_1$ ($A_1 \supseteq A_2$).

The set of concepts from a context ordered by inclusion on the extents forms a complete lattice called the *concept lattice* of the context. Additionally, every complete lattice is the concept lattice of some context. Figure 1.7 (top) shows the concept lattice corresponding to the example of Figure 1.6. One can check that the concepts of Figure 1.7 (top) correspond to maximal boxes of incidence in Figure 1.6. The highlighted concept $(o_2o_7, a_2a_4a_5)$ corresponds to the highlighted concept of Figure 1.6. The concepts that are located directly above (upper cover) and directly below (lower cover) of a concept X form the *conceptual neighborhood* of X.

Two types of concepts can be emphasized. Let o be an object of a formal context. Then, the concept (o'', o') is called an *object-concept*. It is also called the introducer of o. We can state a similar definition for an *attribute-concept*, introducer of the attribute a. For example, in Figure 1.7, the emphasised concept corresponds to the concepts (o_2'', o_2'), and it introduces o_2 in the sense that it is the least concept that contains o_2. In [3], such concepts are denoted respectively $\tilde{\gamma}o$ for object-concepts and $\tilde{\mu}a$ for attribute-concepts. A concept can be both an object-concept and an attribute-concept. Concepts that are neither attribute-concepts nor object-concepts are called plain-concepts. We show some applications of those particular introducer concepts in Chapter 6.

The simplified representation of a concept lattice is a representation where the labels of the concepts are limited. The label for a particular object appears only in the smallest concept that contains it (its introducer). Reversely, the label for a particular attribute appears only in the greatest concept that contains it (its introducer). The other labels are inferred using the inheritance property. Figure 1.7 (bottom) corresponds to the concept lattice from Figure 1.7 (top), with simplified labels. The concept named `Concept_10` has both labels empty. By applying the inheritance, we can retrieve

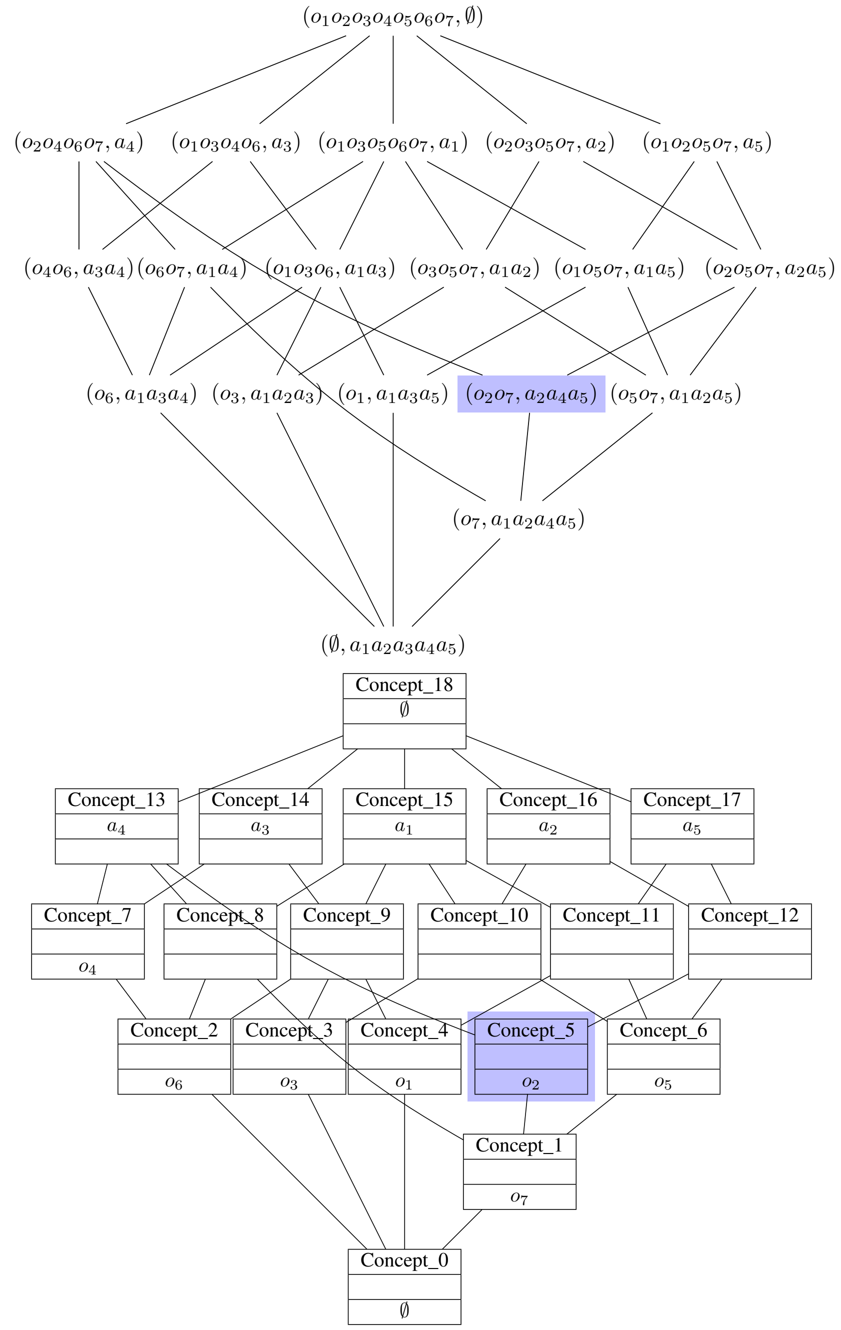


Figure 1.7: Concept lattice corresponding to the example in Fig. 1.6 (top) and with simplified labels (bottom).

that `Concept_10` is in fact the concept $(o_3o_5o_7, a_1a_2)$.

This representation allows a better reading into the concepts (with a clear separation between the extent and the intent of a concept). It allows to see at first glance which concepts are objects-concepts and attribute-concepts (by definition, the introducers have non-empty labels). The plain-concepts are literally plain.

1.4 Polyadic Concept Analysis

Polyadic Concept Analysis is a natural generalisation of FCA. It has been introduced firstly by Lehmann and Wille [4, 5] in the triadic case, and then generalized by Voutsadakis [6].

In Polyadic Concept Analysis, the underlying relation is not binary anymore. Polyadic Concept Analysis deals with d-ary relations between sets. More formally, a d-context can be defined in the following way.

Definition 3. *A d-*context *is a $d+1$ tuple $\mathcal{C} = (\mathcal{S}_1, \dots, \mathcal{S}_d, \mathcal{R})$ where the $\mathcal{S}_i, i \in \{1, \dots, d\}$ are sets called the* dimensions *and $\mathcal{R}$ is a d-ary relation between them.*

A d-context can be represented as a $|\mathcal{S}_1| \times \cdots \times |\mathcal{S}_d|$ crosstable, as shown in Figure 1.8. For technical reasons, most of our examples figures will be drawn in two or three dimensions.

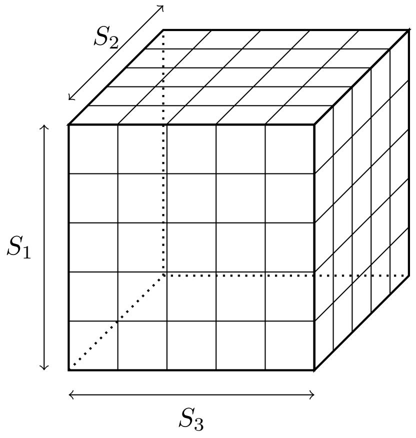


Figure 1.8: Visual representation of a 3-context without its crosses.

When needed, one can represent a d-context by separating the “slices” of the crosstable. For instance, Fig 4.1 shows a 3-context which dimensions are $Numbers$, $Latin$, and $Greek$.

	a	b	c	a	b	c	a	b	c
α	×	×		×			×		
β	×			×			×	×	
γ	×			×		×			×
		1			2			3	

Figure 1.9: A 3-context $\mathcal{C} = (Numbers, Greek, Latin, \mathcal{R})$ where $Numbers = \{1,2,3\}$, $Greek = \{\alpha,\beta,\gamma\}$ and $Latin = \{a,b,c\}$. The relation $\mathcal{R}$ is the set of crosses, that is $\{(1,\alpha,a),(1,\alpha,b),(1,\beta,a),(1,\gamma,a),(2,\alpha,a),(2,\beta,a),(2,\gamma,a),(2,\gamma,c),(3,\alpha,a),$ $(3,\beta,a),(3,\beta,b),(3,\gamma,c)\}$.

The following notations are borrowed from [6]. A d-context gives rise to numerous [3] k-contexts, $k \in \{2,\dots,d\}$. Those k-contexts correspond to partitions π of $\{1,\dots,d\}$ in k subsets such that $\pi = (\pi_1,\dots,\pi_k)$. The k-context corresponding to π is then $\mathcal{C}^\pi = (\prod_{i\in\pi_1} S_i,\dots,\prod_{i\in\pi_d} S_i,\mathcal{R}^\pi)$, where $(s^{(1)},\dots,s^{(k)}) \in \mathcal{R}^\pi$ if and only if $(s_1,\dots,s_d) \in R$ with $s_i \in s^{(j)} \Leftrightarrow i \in \pi_j$.

The contexts $\mathcal{C}^\pi$ are essentially the context $\mathcal{C}$ flattened by merging dimensions with the Cartesian product. A flattened version of our example context $\mathcal{C}$ is shown in Figure 1.10. With every binary partition π comes the associated derivation operator $X \mapsto X^{(\pi)}$.

	(1,a)	(1,b)	(1,c)	(2,a)	(2,b)	(2,c)	(3,a)	(3,b)	(3,c)
α	×	×		×			×		
β	×			×			×	×	
γ	×			×		×			×

Figure 1.10: Let $\pi = (\{Greek\},\{Number, Latin\})$. This figure represents the 2-context $\mathcal{C}^\pi$.

Given a set $D \subseteq \{1,\dots,d\}$ of dimensions, $\overline{D} = \{1,\dots,d\}\backslash D$. We denote by $\mathcal{S}_D$ the collection $\mathcal{S}_D = \langle \mathcal{S}_d \mid i \in D\rangle$. Let $X_i \subseteq \mathcal{S}_d$ be a set of elements of dimension i, we denote by $X_D = \langle X_i \mid i \in D\rangle$ and x_D the elements of $\prod_{i\in D}\mathcal{S}_i$. The $|\overline{D}|$-context associated with X_D is $\mathcal{C}_{X_D} = (\mathcal{S}_{\overline{D}}, R_{X_D})$ such that $\forall x_{\overline{D}} \in \prod_{i\in\overline{D}}\mathcal{S}_i, x_{\overline{D}} \in R_{X_D}$ if and only if $x_{D\cup\overline{D}} \in R$.

The contexts C_{X_D} correspond to the intersection of the flattened versions, for given subsets on some dimensions, as shown in Figure 1.11.

	a	b	c
α	×		
β	×		
γ			

	a	b	c
1	×	×	
2	×		
3	×		

Figure 1.11: 2-Contexts $\mathcal{C}_{\langle\{1,3\}\rangle}$ (left) and $\mathcal{C}_{\langle\{\alpha\}\rangle}$ (right).

[3] Stirling number of the second kind, or number of ways of arranging d dimensions into k slots.

The pair sets $\overline{D} = \{d_1, d_2\}$ and the sets $X_d \subseteq S_i, i \in D$ give rise to the derivation operation $X \mapsto X^{(d_1,d_2,X_D)}$ on the 2-context $\mathcal{C}_{X_D}^{\{d_1,d_2\}}$.

In the same way as in the 2 dimensional case, d-dimensional maximal boxes of incidence have an important role in data mining. The d-dimensional maximal boxes of incidence in a d-context are called d-concepts. We will usually denote d-dimensional concepts as d-concepts, except when the dimension is clear from the context and we will simply denote them concepts.

Definition 4. *A d-*concept *of $\mathcal{C} = (\mathcal{S}_1, \ldots, \mathcal{S}_d, \mathcal{R})$ is a d-tuple $(X_1, \ldots, X_d)$ such that $\prod_{i \in \{1,\ldots,d\}} X_i \subseteq \mathcal{R}$ and, for all $i \in \{1, \ldots, d\}$, there is no $k \in \mathcal{S}_i \setminus X_i$ such that $\{k\} \times \prod_{j \in \{1,\ldots,d\} \setminus \{i\}} X_j \subseteq R$.*

A d-concept of a d-context $(\mathcal{S}_1, \ldots, \mathcal{S}_d, \mathcal{R})$ is a d-tuple $(X_1, \ldots, X_d)$ such that $X_i \subseteq \mathcal{S}_i$ for all $i \in \{1, \ldots, d\}$ and $X_i = X_{\{1,\ldots,d\}\setminus\{i\}}^{(i,\{1,\ldots,d\}\setminus\{i\})}$. As in the 2-dimensional case, d-concepts can be defined with respect to the derivation operators. However, their definition as maximal boxes of incidence is more convenient, and will be used much more in this dissertation.

In the 2-dimensional case, 2-concepts can be ordered by the subset inclusion on either objects or attributes. The structure that results from this ordering is a partial order with some properties that makes it a lattice. In the d-dimensional case, d-concepts can be ordered in a slighty different way.

Definition 5. *$\mathcal{P} = (P, \lesssim_1, \ldots, \lesssim_d)$ is a d-ordered set if for $A \in P$ and $B \in P$:*

1. *$A \sim_i B, \forall i \in \{1, \ldots, d\}$ implies $A = B$ (Uniqueness Condition)*
2. *$A \lesssim_{i_1} B, \ldots, A \lesssim_{i_{d-1}} B$ implies $B \lesssim_{i_d} A$ (Antiordinal Dependency)*

Here, the $\lesssim_i$ are quasi-orders, binary relations that are reflexive and transitive. For the quasi-orders $\lesssim_i, i \in \{1, \ldots, d\}$, the d-concepts of a d-context $\mathcal{C}$ can be ordered in the following way:

$$(A_1, \ldots, A_d) \lesssim_i (B_1, \ldots, B_d) \Leftrightarrow A_i \subseteq B_i$$

The equivalent relation $\sim_i, i \in \{1, \ldots, d\}$ is defined in the same way:

$$(A_1, \ldots, A_n) \sim_i (B_1, \ldots, B_n) \Leftrightarrow A_i = B_i$$

We can see that concept lattices are in fact 2-lattices that respect both the uniqueness condition and the antiordinal dependency. They are isomorphic to 1-lattices because a concept is uniquely defined by its intent or its extent, and so the two orders are dual.

From now on, we will use the notation $\{ab\}$ to denote the set $\{a, b\}$.

In order to fully understand d-ordered sets, let us illustrate the definition with a small digression about graphical representation. In 2 dimensions, for concept lattices, the two orders (on the extent and on the intent) are dual and only one is usually mentioned (the set of concepts ordered by inclusion on the extent, for example). Thus, their representation is possible with Hasse diagrams. From dimension 3 and up, the

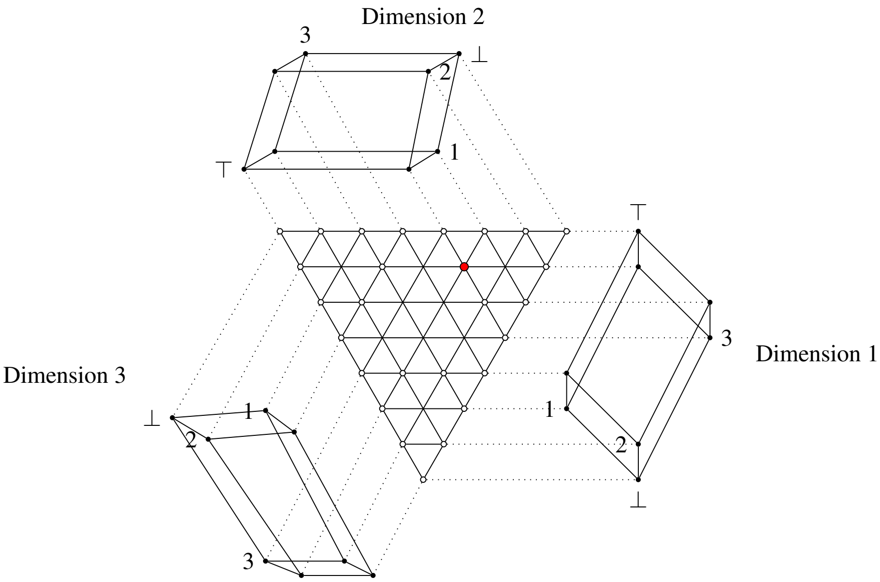


Figure 1.12: This represents a powerset 3-lattice. The red concept is $(13, 13, 23)$.

representation of d-ordered sets is harder. For example, in dimension 3, no good (in the sense that it allows to represent any 3-ordered set) representation exists. However there is still a possible representation in the form of triadic diagrams.

Figure 1.12 shows an example of a *triadic diagram*, a representation of a 3-ordered set. Let us explain how this diagram should be read. The white circles in the central triangle are the concepts. The lines of the triangle represent the equivalent relation between concepts: the horizontal lines represent $\sim_1$, the north-west to south-east lines represent $\sim_2$ and the south-west to north-east lines represent $\sim_3$. Recall that two concepts are equivalent with respect to $\sim_i$ if they have the same coordinate on dimension i. This coordinate can be read by following the dotted line until the diagrams outside the triangle. In [4], Lehmann and Wille call the external diagrams the *extent diagram*, *intent diagram* and *modi diagram* depending on which dimension they represent. Here, to pursue the permutability of the dimensions further, we simply denote them by *dimension diagram* for dimension 1, 2 or 3.

If we position ourselves on the red concept of Figure 1.12, we can follow the dotted line up to the dimension 1 diagram and read its coordinate on the first dimension. It is $\{13\}$. Following the same logic for dimension 2 and 3, we can see that the red concept is $(13, 12, 23)$. The concept on the south-east of the red concept is on the same equivalent line. We know that its second dimension will be $\{12\}$ too. When we follow the lines to the dimension diagrams, we see that it is $(23, 12, 13)$.

Here, every dimension diagram is a powerset on 3 elements. It is not always the case, as the three dimension diagrams are not always isomorphic.

We mentioned earlier that this representation does not allow to draw any 3-ordered set (with straight lines). In [4], Lehmann and Wille explain this by saying that a violation of the so-called 'Thomsen Conditions [7]' and their ordinal generalisation [8] may appear in triadic contexts.[4].

After this digression, let us now dive into the particular joins that make a d-ordered set a d-lattice. Let $\mathcal{P} = (P, \lesssim_1, \dots, \lesssim_d)$ be a d-ordered set, $j_1, j_2, \dots, j_{d-1} \in \{1, \dots, d\}$ be indexes and $X_1, X_2, \dots, X_{d-1}$ be some elements of P.

Definition 6. *An element $p \in P$ is called a $(j_{d-1}, \dots, j_1)$-*bound *of $(X_{d-1}, \dots, X_1)$ if $x_i \lesssim_{j_i} p$ for all $x_i \in X_i, i \in \{1, \dots, d-1\}$. The set of all $(j_{d-1}, \dots, j_1)$-bounds of $(X_{d-1}, \dots, X_1)$ is denoted by $(X_{d-1}, \dots, X_1)^{(j_{d-1}, \dots, j_1)}$*

Such a bound l is called a $(j_{d-1}, \dots, j_1)$-limit of $(X_{d-1}, \dots, X_1)$ if $b \lesssim_{j_d} l$ for all $(j_{d-1}, \dots, j_1)$-bound b of $(X_{d-1}, \dots, X_1)^{(j_{d-1}, \dots, j_1)}$.

This proposition is Proposition 4 from [6].

Proposition 7 ([6, Proposition 4])**.** *Let $\mathcal{P} = (P, \lesssim_1, \dots, \lesssim_d)$ be an d-ordered set, $X_1, \dots, X_{d-1} \subseteq P$ elements of this set and $\{j_1, \dots, j_d\} = \{1, \dots, d\}$ an index set. Then there exists at most one $(j_{d-1}, \dots, j_1)$-limit $\bar{l}$ of $(X_{d-1}, \dots, X_1)$ satisfying "$\bar{l}$ is the largest in $\lesssim_{j_2}$ among the largest limits in $\lesssim_{j_3}$ among $\dots$ among the largest limits in $\lesssim_{j_{d-1}}$ among the largest limits in $\lesssim_{j_d}$".*

Such a limit is called the $(j_{d-1}, \dots, j_1)$-*join* of $(X_{d-1}, \dots, X_1)$ and is denoted $\nabla_{j_{d-1}, \dots, j_1}(X_{d-1}, \dots, X_1)$. A complete d-lattice $\mathbb{L} = (L, \lesssim_1, \dots, \lesssim_d)$ is a d-ordered set in which all $\nabla_{j_{d-1}, \dots, j_1}(X_{d-1}, \dots, X_1)$-joins exist, for all $(X_{d-1}, \dots, X_1) \subseteq L$ and all $\{j_1, \dots, j_d\} = \{1, \dots, d\}$.

The basic theorem of Polyadic Concept Analysis [6] shows that the set of d-concepts of a d-context together with the d quasi-orders induced by the inclusion relation on the subsets of each dimensions forms a complete d-lattice. Additionally, every complete d-lattice is isomorphic to the one formed by the d-concepts of a particular d-context. This makes Figure 1.12 a 3-lattice.

Conclusion

In this chapter we introduced the central notions of this thesis. We started from the rather large notion of partially ordered sets, only to reduce it to the class of lattices. The condensed representation of lattices into binary relations (Formal Concept Analysis) allows to study some aspects of lattices by studying the formal contexts, and thus to work on a smaller object. From a binary to a d-ary relation, there was just a step, that we took gladly, and introduced d-ordered sets, and from that, d-lattices.

Now that we have reduced the scope from "theoretical computer science" to order theory, lattices and d-lattices, the reader has most of the necessary definitions and knowledge to ensure a smooth reading of the rest of this dissertation. Of course, we will meet some additional notions, and their definitions will be delivered in due time.

[4] I was unable to put my hands on those two books in order to learn more about that. If you have a copy, I am most interested.

Part I

Combinatorics

“As I was going to St. Ives,
I met a man with seven wives,
Each wife had seven sacks,
Each sack had seven cats,
Each cat had seven kits,
Kits, cats, sacks and wives,
How many were going to St. Ives?
POPULAR RHYME (XVII CENTURY)

“Seven old women are going to Rome,
Each of them has seven mules,
Each mule carries seven sacks,
Each sack contains seven loaves,
Each loaf has seven knives,
And each knife has seven sheaths.
What is the total number of things?
LEONARDO FIBONACCI (XIII CENTURY)

House	*7*
Cats	*49*
Mice	*343*
Wheat	*2401*
Hekat	*16807*
	19607

RHIND PAPYRUS (XVII CENTURY BC)

Questions about combinations, permutations and counting things in general go a long way back in history. In both eastern and western mathematics, one can find many examples of combinatorial problems through the ages. The three examples above show that some basic problems have even gone into the popular culture. These three examples, and a more general survey about early ages combinatorics can be found in [9].

When dealing with interesting structures such as lattices and d-lattices, one of the first questions one can ask is about their cardinality. In this part, we study the cardinality of d-lattices. More precisely, in the first part we focus on the maximal size of a d-lattice, to try to answer the question “how big can a d-lattice be”. We study the maximal size of a 3-lattice and give some insight about the general, d-dimensional case. In the second chapter, we study the average case combinatorics of different mathematical objects : implications between elements of a set, and d-concepts. Sure d-lattices can get big, but fear not, the average case is quasi-nice.

Chapter 2

The extremal case: how big can a d-lattice get?

In this chapter, we are interested in the number of concepts that arise in some contexts. In particular, we are interested in the maximal number of 3-concepts that can appear in 3-contexts. We give an upper bound and a lower bound on this number. We also discuss powerset d-lattices and give some insight about their construction and their number of concepts.

2.1 d-Lattices as Hypergraphs

In this section, we stray from d-lattices and their order properties to try and visualise d-contexts as hypergraphs. The transformation from d-context to hypergraphs allows us to handle simpler objects. The proofs that we present in this chapter are mostly formalised in terms of hypergraphs. In order to have a smooth reading of this chapter, we recall some definitions and properties of hypergraphs.

Hypergraphs are a generalization of graphs, where edges may have arity different than 2. They have been formalized by Berge in the seventies [10]. Formally, a *hypergraph* $\mathcal{H}$ is a pair $(V, \mathcal{E})$, where V is a set of vertices and $\mathcal{E}$ is a family of subsets of V called hyperedges. We suppose that $V = \bigcup \mathcal{E}$. From now on, we may use edge in place of hyperedge. The notations $V(\mathcal{H})$ and $\mathcal{E}(\mathcal{H})$ denote respectively the set of vertices and the set of edges of a hypergraph $\mathcal{H}$. The order of a hypergraph is its number of vertices. When all the hyperedges of a hypergraph have the same arity k, we call it a *k-uniform hypergraph*. Graphs are exactly the 2-uniform hypergraphs. When the set of vertices of a hypergraph can be partitioned into k sets, such that every edge intersects each part at most once, the hypergraph is called *k-partite*.

Let n, d and k be integers, such that $1 < d < n$ and $1 < k < n$. Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph of order n. A *k-clique* or *k-complete subgraph* of $\mathcal{H}$ is a set S of vertices such that all the subsets of S of size k are edges. A *d-partite complete subgraph* of a d-partite hypergraph is a subset S of its vertices such that every subset of S of size d that has an element from each part forms an edge. A *stable* or *independent set* in a hypergraph is a set S of vertices that contains no edge.

Of course, one can infer a definition for a *maximal* k-clique, *maximal* d-partite complete subgraph or *maximal* stable by simply stating that no superset of the solution has the same property.

A *transversal* of $\mathcal{H}$ is a set of vertices that intersects every edge of $\mathcal{H}$. In the same way as before, a minimal transversal is a transversal that does not contain another transversal as one of its subsets. Formally, $S \subseteq V$ is a transversal of $\mathcal{H} = (V, \mathcal{E})$ if $S \cap E \neq \emptyset, \forall E \in \mathcal{E}$. It is a minimal transversal if there is no $S' \subset S$ such that S' is a transversal of $\mathcal{H}$.

Armed with those basic definitions, we can head into the more interesting part: d-contexts as hypergraphs.

2.1.1 d-Contexts as Hypergraphs

Binary datasets, which include d-contexts, are essentially hypergraphs. The hypergraph $\mathcal{H}_\mathcal{C}$ associated with the d-context $\mathcal{C} = (\mathcal{S}_1, \ldots, \mathcal{S}_d, \mathcal{R})$ is the hypergraph in which each edge represents an element of the relation $\mathcal{R}$ (basically a cross of the context).

Definition 8. *The hypergraph associated with a d-context $\mathcal{C}$ is the hypergraph $\mathcal{H}_\mathcal{C} = (V, \mathcal{E})$ where $V = \bigcup_{i \in \{1,\ldots,d\}} \mathcal{S}_i$ and $\mathcal{E} = \{\{x_1, \ldots, x_d\} \mid (x_1, \ldots, x_d) \in \mathcal{R}\}$.*

For example, in a 2-dimensional setting, a context is a bipartite graph, and its concepts are maximal complete bipartite subgraphs (subgraphs in which every pair of vertices that does not belong to the same part is an edge). In the 3-dimensional case, a 3-context is a tripartite, 3-uniform hypergraph and its 3-concepts are then complete tripartite sub-hypergraphs (subgraph in which every three vertices that does not belong to the same part form an edge).

In Figure 2.1 and Figure 2.2 we show a 2-context and its associated 2-uniform hypergraph. With $\mathcal{C}$ a d-context, the set of d-concepts of $\mathcal{C}$ is the set of maximal complete d-partite subgraphs of the corresponding d-partite, d-uniform hypergraph.

	a	b	c	d
1	×		×	
2		×		×
3	×	×	×	
4	×		×	×

Figure 2.1: A 2-context with 4 objects and 4 attributes. The concept $(134, ac)$ is highlighted.

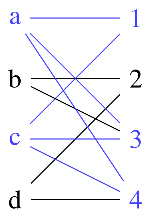


Figure 2.2: The 2-uniform hypergraph that correspond to the 2-context on the left. The maximal complete bipartite subgraph $(134, ac)$ is highlighted.

	a	b	c	d
1	×		×	
2		×		×
3	×	×	×	
4	×		×	×

Figure 2.3: A 2-context with 4 objects and 4 attributes. The concept $(134, ac)$ is highlighted.

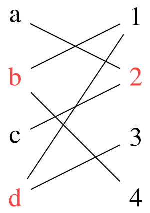


Figure 2.4: The complement hypergraph of the 2-context on the left. Here, the set $\{bd2\}$ is a minimal transversal of the hypergraph.

2.1.2 Changing paradigms: hypergraph transversals

In this section, we change paradigm and focus on hypergraph transversals.

We call complement hypergraph of a context $\mathcal{C}$ (and denote it $\overline{\mathcal{H}_\mathcal{C}}$) the hypergraph obtained by creating a hyperedge for each cell that does not contain a cross in $\mathcal{C}$.

Definition 9. *The* complement hypergraph $\overline{\mathcal{H}_\mathcal{C}}$ *of a d-context $\mathcal{C}$ is the hypergraph $\overline{\mathcal{H}_\mathcal{C}} = (V, \mathcal{E})$ where $V = \bigcup_{i \in \{1,\ldots,d\}} \mathcal{S}_i$ and*

$$\mathcal{E} = \{\{x_1, \ldots, x_d\} | (x_1, \ldots, x_d) \in \prod_{i \in \{1,\ldots,d\}} \mathcal{S}_i \setminus \mathcal{R}\}.$$

In either the hypergraph associated with an d-context or its complement hypergraph, all edges have arity d (as they represent the coordinates of either the crosses or the holes). They are also d-partite, since there is no edge containing two elements of a same dimension. In Figure 2.3, we show a 2-context and its complement bipartite graph (Figure 2.4).

A d-concept of $\mathcal{H}_\mathcal{C}$ is a subset S of the vertices of $\mathcal{H}_\mathcal{C}$ such that every d-tuple of vertices from different parts forms an edge and that cannot be augmented with respect to this property, otherwise known as a maximal complete d-partite subgraph. In $\overline{\mathcal{H}_\mathcal{C}}$, the same subset S of vertices (recall that $\mathcal{H}_\mathcal{C}$ and $\overline{\mathcal{H}_\mathcal{C}}$ have the same vertex set) is a maximal stable set. Now, still in $\overline{\mathcal{H}_\mathcal{C}}$, $S' = V(\overline{\mathcal{H}_\mathcal{C}}) \setminus S$ is a minimal transversal. Indeed, suppose that not all the edges are intersected by S', then there is an edge E that contains no element of S'. This implies that all the elements of E are in S, and that is a contradiction with the fact that S is a stable set. Now suppose that S' is not minimal. There exists an element x of S' such that $S' \setminus x$ is a transversal. This implies that $S \cup \{x\}$ is a stable, that is a contradiction with the fact that S is maximal.

Proposition 10. *Let $(X_1, \ldots, X_d)$ be a d-concept of a d-context $\mathcal{C}$. Then*

$$\prod_{i \in \{1,\ldots,d\}} \mathcal{S}_i \setminus \prod_{i \in \{1,\ldots,d\}} X_i$$

is a minimal transversal of $\overline{\mathcal{H}_\mathcal{C}}$.

This property allows us to shift paradigm and consider minimal transversal instead of concepts. In Figure 2.4, the set of vertices that corresponds to a concept is a stable set, and its complement with respect to $V(\overline{\mathcal{H}_\mathcal{C}})$ is a minimal transversal of $\overline{\mathcal{H}_\mathcal{C}}$. Additionally, minimal transversals of hypergraphs are equivalent to other notions of databases and data mining (see [11, 12] and references therein) and are well worth studying even without this equivalence to concepts. This equivalence is well known in formal concept analysis.

This shift allows us to explore some structural properties of hypergraphs in the next section. Also, as a good friend of mine said: “better to search for sparse things: transversals are clearer than cliques”.[1]

2.2 An Upper Bound on the Size of a 3-Lattice

In order to study the maximal size of a 3-lattice, we shift to minimal transversals in tripartite 3-uniform hypergraphs. In this whole chapter, we denote by n the order of the hypergraphs. This is different from the classical notation of formal concept analysis, where n is usually the size of only one dimension. [2] Here, since we consider the whole hypergraph, $n = \sum_{i \in \{1,\ldots,d\}} |\mathcal{S}_i|$.

We are interested in the number of minimal transversals that arise in tripartite 3-uniform hypergraphs. We denote by $f_3(n)$ the maximum number of minimal transversal in such hypergraphs of order n.

By fixing the vertices taken in two of the three sets of vertices, there is only one way to pick the remaining vertices. This is equivalent to the unicity of a 3-concept described by two out of three components. A naive bound is then found by assuming that all subsets from two of the three sets of vertices can appear in minimal transversals, which gives an upper bound of $2^{n/3} \times 2^{n/3} = 4^{n/3} \approx 1.5874^n$ minimal transversals. That is a bold assumption, as we will show in this section that $f_3(n)$ is in fact bounded by 1.5012^n.

In this section, we prove the following theorem.

Theorem 11. *For any integer n, $f_3(n) \leq 1.5012^n$.*

[1] Credit where credit is due, I think that this sentence originated from a discussion with Laurent Beaudou and Jérémie Chalopin, in Lyon.

[2] See for example the commonplace statement that a 2-context has at most 2^n concepts.

2.2.1 Proof of Theorem 11

The proof of Theorem 11 relies on a technique introduced by Kullman [13] and used on similar objects by Lonc and Truszczyński [14]. It resembles the approach used in the analysis of exact exponential-time algorithms, *measure and conquer* by Fomin, Grandoni and Kratsch [15].

In [14], the authors consider the class of hypergraphs of rank 3 (hypergraphs in which the edge have arity at most 3). This class contains the tripartite 3-uniform hypergraphs, but the general bound they obtained (1.6702^n) is larger than the trivial one in our specific case.

We define a rooted tree by forcing the presence or absence of some vertices in a minimal transversal. In this tree, each internal node is a hypergraph from our family, and each leaf corresponds to a minimal transversal, as shown in Figure 2.5. The next step is to count the number of leaves in this tree. We do that by using the so-called "τ-lemma" introduced by Kullmann [13, Lemma 8.2]. We associate a measure to each hypergraph of our class. Then, using this measure, we label the edges of our tree with a carefully chosen distance between the parent (hypergraph) and the child (hypergraph).

The estimation of the number of leaves is then done by computing the maximal τ, that depends on the measure for each hypergraph (for each inner node of the tree). This constitutes the basis of the exponential. The exponent is the maximal distance from the root of the tree to its leaves.

Let us dive into the more technical part. We mainly use the notion of *condition*, introduced thereafter.

Definition 12. *Given a set V of vertices, a* condition *on V is a pair (A^+, A^-) of disjoint subsets of V. A condition is* trivial *if $A^+ \cup A^- = \emptyset$, and* non-trivial *otherwise.*

All the conditions that we handle are non-trivial. A condition amounts to fixing a set of vertices to be part of the solution (the vertices in A^+) and forbidding other vertices (the vertices in A^-).

A set T of vertices *satisfies* a condition (A^+, A^-) if $A^+ \subseteq T$ and $T \cap A^- = \emptyset$.

Let $\mathcal{H}$ be a hypergraph and (A^+, A^-) a condition on the vertices of $\mathcal{H}$. The hypergraph $\mathcal{H}_{(A^+,A^-)}$ is obtained from $\mathcal{H}$ and (A^+, A^-) using the following procedure:

1. remove every edge that contains a vertex that is in A^+ (as we take the vertices of A^+, this edge is now covered);

2. remove from every remaining edge the vertices that are in A^- (we prohibit the vertices from A^-);

3. remove redundant edges (as they have the same transversals, we do not keep the duplicates).

Vertices of $\mathcal{H}$ that appear in a condition (A^+, A^-) are not in $V(\mathcal{H}_{(A^+,A^-)})$ as they are either removed from all the edges or all the edges that contained them have disappeared.

Lemma 13. *Let $\mathcal{H}$ be a hypergraph, (A^+, A^-) a condition and T a set of vertices of $\mathcal{H}$. If T is a minimal transversal of $\mathcal{H}$ and T satisfies (A^+, A^-), then $T \setminus A^+$ is a minimal transversal of $\mathcal{H}_{(A^+,A^-)}$.*

Proof. The proof is straightforward from the construction of $\mathcal{H}_{(A^+,A^-)}$. □

This echoes the parent-child relation that can be found in the exact exponential-time algorithm field [16].

A family of conditions is *complete* for the hypergraph $\mathcal{H}$ if the family is non-empty, each condition in the family is non-trivial, and every transversal of $\mathcal{H}$ satisfies at least one condition of the family.

Let $\mathcal{C}$ be the class of tripartite hypergraphs in which edges have arity at most 3 and one of the parts is a minimal transversal. We call S the part that is a minimal transversal. Tripartite 3-uniform hypergraphs belong to this class. Tripartite hypergraphs remain tripartite when vertices are removed from edges or edges are deleted. Of course, those operations do not increase the arity of edges. As such, if an hypergraph $\mathcal{H} = (V, \mathcal{E})$ belongs to the class $\mathcal{C}$ and $A = (A^+, A^-)$ is a condition on V such that the edges that contain vertices of $A^- \cap S$ also contain vertices of A^+, then $\mathcal{H}_A$ is in $\mathcal{C}$. From now on, we suppose that all the conditions we discuss respect this property.

A hypergraph is non-trivial if it is not empty. A *descendant function* for $\mathcal{C}$ is a function that assigns to each non-trivial hypergraph in $\mathcal{C}$ a complete family of conditions. Let ρ be such a function.

We define a rooted labeled tree $\mathcal{T}_\mathcal{H}$ for all hypergraphs $\mathcal{H}$ in $\mathcal{C}$. If $\mathcal{H} = \emptyset$, then we define $\mathcal{T}_\mathcal{H}$ to be a single node labeled with $\mathcal{H}$. When $\mathcal{H}$ is non-trivial, we create a node labeled $\mathcal{H}$ and make it the parent of all trees $\mathcal{T}_{\mathcal{H}_A}$ for all $A \in \rho(\mathcal{H})$. Since $\mathcal{C}$ is closed under the operation of removing edges and removing vertices from edges and since the number of vertices can only decrease when the transformation from $\mathcal{H}$ to $\mathcal{H}_A$ occurs, the inductive definition is valid. An example of such a tree is shown in Figure 2.5.

Proposition 14. *Let ρ be a descendant function for a class closed by removing edges and removing vertices from edges. Then for all hypergraphs $\mathcal{H}$ in such a class, the number of minimal transversals is bounded above by the number of leaves of $\mathcal{T}_\mathcal{H}$.*

Proof. When $\mathcal{H} = \emptyset$, then it has only one transversal, the empty set. If the empty set is an edge of $\mathcal{H}$, then $\mathcal{H}$ has no transversals. In both cases, the assertion follows directly from the definition of the tree. Let us assume now that $\mathcal{H}$ is non-trivial, and that the assertion is true for all hypergraphs with fewer vertices than $\mathcal{H}$.

As $\mathcal{H}$ is non-trivial, ρ is well defined for $\mathcal{H}$ and gives a complete family of conditions. Let X be a minimal transversal of $\mathcal{H}$. Then X satisfies at least one condition A in $\rho(\mathcal{H})$. From Lemma 13, we know that there is a minimal transversal Y of $\mathcal{H}_A$ such that $Y \cup A^+ = X$. Then the number of minimal transversals of $\mathcal{H}$ is at most the sum of the number of minimal transversals in its children. □

We now want to bound the number of leaves in $\mathcal{T}_\mathcal{H}$ for all hypergraphs in $\mathcal{C}$. To that end, we shall use Lemma 15 proved by Kullmann [13].

We denote by $L(\mathcal{T})$ the set of leaves of $\mathcal{T}$ and for a leaf ℓ, we denote by $P(\ell)$ the set of edges on the path from the root to leaf ℓ.

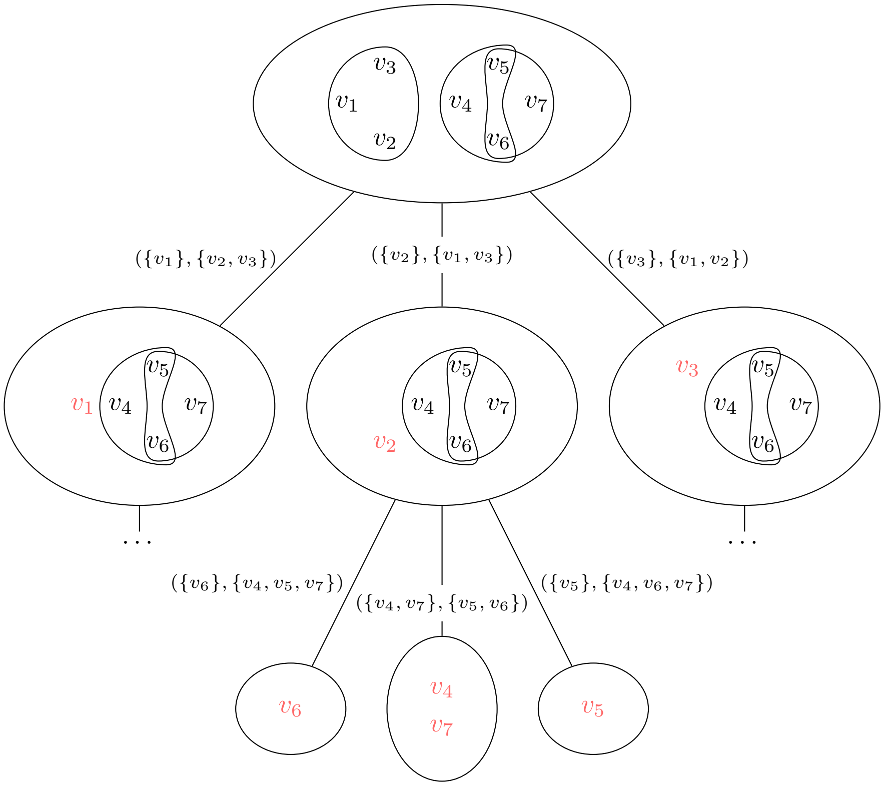

v_3
v_1
v_2
v_5
v_4
v_7
v_6
$(\{v_1\},\{v_2,v_3\})$
$(\{v_2\},\{v_1,v_3\})$
$(\{v_3\},\{v_1,v_2\})$
v_1 v_4 v_5 v_7 v_6
v_2 v_4 v_5 v_7 v_6
v_3 v_4 v_5 v_7 v_6
...
...
$(\{v_6\},\{v_4,v_5,v_7\})$
$(\{v_4,v_7\},\{v_5,v_6\})$
$(\{v_5\},\{v_4,v_6,v_7\})$
v_6
v_4
v_7
v_5

Figure 2.5: Example of a tree constructed from some hypergraphs. Each leaf corresponds to a minimal transversal, so bounding the number of leaves gives a bound on the number of minimal transversals. A transversal can be read by reading the red labels (vertices of A^+) going up. The tree is not fully drawn.

Lemma 15 ([13, Lemma 8.1])**.** *Consider a rooted tree $\mathcal{T}$ with an edge labeling w with value in the interval $[0,1]$ such that for every internal node, the sum of the labels on the edges from that node to its children is 1 (that is a transition probability).*

Then,

$$|L(\mathcal{T})| \leq \max_{\ell \in L(\mathcal{T})} \left(\prod_{e \in P(\ell)} w(e) \right)^{-1} .$$

An example of how this lemma works can be found in Figure 2.6 and Figure 2.7. We give two example of probability transitions and the bound that they imply for a given tree. This shows that we want to find a probability transition that maximize the less probable leaf in our tree.

In order to pick an adequate probability distribution, we use a measure. In our context, a *measure* is a function that assigns to any hypergraph $\mathcal{H}$ in $\mathcal{C}$ a real number $\mu(\mathcal{H})$ such that $0 \leq \mu(\mathcal{H}) \leq |V(\mathcal{H})|$. Let $\mathcal{H}$ be such a hypergraph, A a condition on

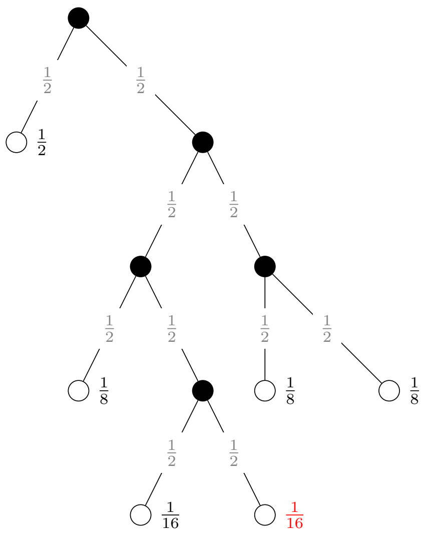

$\frac{1}{2}$ $\frac{1}{2}$
$\frac{1}{2}$
$\frac{1}{2}$ $\frac{1}{2}$
$\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$
$\frac{1}{8}$ $\frac{1}{8}$ $\frac{1}{8}$
$\frac{1}{2}$ $\frac{1}{2}$
$\frac{1}{16}$ $\frac{1}{16}$

Figure 2.6: The (most obvious) transition probability given on the edges of this tree gives us that the least probable leaf has probability $\frac{1}{16}$. This implies that this tree has is number of leaves bounded by 16.

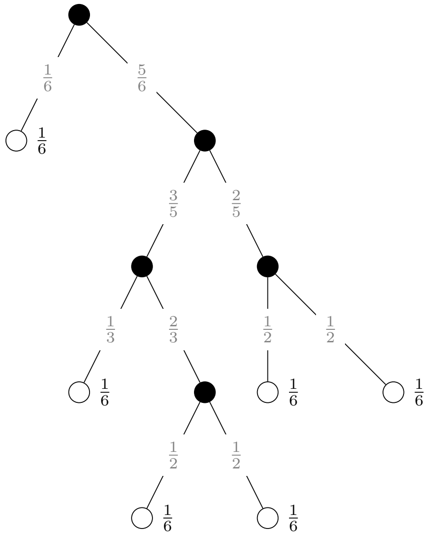

$\frac{1}{6}$ $\frac{5}{6}$
$\frac{1}{6}$
$\frac{3}{5}$ $\frac{2}{5}$
$\frac{1}{3}$ $\frac{2}{3}$ $\frac{1}{2}$ $\frac{1}{2}$
$\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{6}$
$\frac{1}{2}$ $\frac{1}{2}$
$\frac{1}{6}$ $\frac{1}{6}$

Figure 2.7: This other probability gives a more balanced probability for each leaves. The least probable leaf has probability $\frac{1}{6}$, which ensures that the number of leaves of this tree is bounded by 6.

its vertices and μ a measure. We define

$$\Delta(\mathcal{H}, \mathcal{H}_A) = \mu(\mathcal{H}) - \mu(\mathcal{H}_A).$$

Let $\mathcal{H}$ be a hypergraph in $\mathcal{C}$ and μ a measure. If, for every condition A in $\rho(\mathcal{H})$, $\mu(\mathcal{H}_A) \leq \mu(\mathcal{H})$, that we say then ρ is μ-compatible. In this case, there is a unique positive real number $\tau \geq 1$ such that

$$\sum_{A \in \rho(\mathcal{H})} \tau^{-\Delta(\mathcal{H}, \mathcal{H}_A)} = 1. \tag{2.1}$$

When $\tau \geq 1$, $\sum_{A \in \rho(\mathcal{H})} \tau^{-\Delta(\mathcal{H}, \mathcal{H}_A)}$ is a strictly decreasing continuous function of τ. For $\tau = 1$, it is at least 1, since $\rho(\mathcal{H})$ is not empty, and it tends to 0 when τ tends to infinity.

A descendant function defined on a class $\mathcal{C}$ is μ-bounded by τ_0 if, for every non-trivial hypergraph $\mathcal{H}$ in $\mathcal{C}$, $\tau_{\mathcal{H}} \leq \tau_0$.

Now, we adapt the τ-lemma proven by Kullmann [13] to our formalism.

Theorem 16 (Kullmann [13])**.** *Let μ be a measure and ρ a descendant function, both defined on a class $\mathcal{C}$ of hypergraphs closed under the operation of removing edges and removing vertices from edges. If ρ is μ-compatible and μ-bounded by τ_0, then for every hypergraph $\mathcal{H}$ in $\mathcal{C}$,*

$$|L(\mathcal{T}_{\mathcal{H}})| \leq \tau_0^{h(\mathcal{T}_{\mathcal{H}})}$$

where $h(\mathcal{T}_{\mathcal{H}})$ is the height of $\mathcal{T}_{\mathcal{H}}$.

This theorem comes from Lemma 15. We find the worst possible branching and assume that we apply it on every node on the path from the root to the leaves.

Lemma 17. *There is a measure μ defined for every hypergraph $\mathcal{H}$ in $\mathcal{C}$ and a descendant function ρ for $\mathcal{C}$ that is μ compatible and μ-bounded by* 1.8393.

Proof. The general idea of this proof is that we give a complete family of conditions for every non-trivial hypergraph in $\mathcal{C}$.

Let $\mathcal{H}$ be a hypergraph belonging to the class $\mathcal{C}$, i.e. a tripartite hypergraph that contains a set S of vertices that is a minimal transversal such that no two vertices of S belong to a same edge.

We use Theorem 16 to bound the number of leaves in the tree $\mathcal{T}_{\mathcal{H}}$ and thus the number of minimal transversals in $\mathcal{H}$. To do so, we define a descendant function ρ that assigns a family of conditions to $\mathcal{H}$ depending on its structure. This takes the form of a case analysis.

In each case, we consider a vertex a from S and its surroundings. The conditions involve vertices from these surroundings and are sometimes strengthened to contain a if and only if its presence in A^+ or A^- is implied by our hypotheses. This causes every condition A to respect the property that A^+ intersects all the edges containing a vertex of $A^- \cap S$, which lets $\mathcal{H}_A$ remain in the class $\mathcal{C}$.

We set the measure $\mu(\mathcal{H})$ to

$$\mu(\mathcal{H}) = |V(\mathcal{H})| - \alpha m(\mathcal{H})$$

where $m(\mathcal{H})$ is the maximum number of pairwise disjoint 2-element edges in $\mathcal{H}$ and $\alpha = 0.145785$. The same measure is used in [14] (with a different α).[3]

In each case i and for every condition A, we find a bound $k_{\mathcal{H},A}$ such that

$$k_{\mathcal{H},A} \leq \Delta(\mathcal{H}, \mathcal{H}_A)$$

and a unique positive real number τ_i that satisfies the equation

$$\sum_{A \in \rho(\mathcal{H})} \tau_i^{-k_{\mathcal{H},A}} = 1$$

We show that $\tau_i \leq 1.8393$ for all i. Let τ_0 represent the quantity 1.8393.

As all our conditions involve at least one element from $V(\mathcal{H}) \setminus S$, the height of $\mathcal{T}_{\mathcal{H}}$ is bounded by $|V(\mathcal{H})| - |S|$. Hence, we have

$$|L(\mathcal{T}_{\mathcal{H}})| \leq \tau_0^{|V(\mathcal{H})| - |S|}$$

In the remainder of the proof, we will write conditions as sets of expressions of the form a and $\overline{b}$ where a means that a is in A^+ and $\overline{b}$ means that b is in A^-. For example, the condition $(\{a, c\}, \{b, d, e\})$ will be denoted $ac\overline{b}\overline{d}\overline{e}$. For a vertex v, we denote by $d_2(v)$ the number of 2-edges that contain v, and by $d_3(v)$ the number of 3-edges that contain v.

[3] The α is defined experimentally in order to optimize the end result.

Case 1: $d_2(a) \geq 2$: the hypergraph $\mathcal{H}$ contains a vertex a from S that belongs to no 3-edge and at least two 2-edges ab and ac.

A minimal transversal of $\mathcal{H}$ either contains or does not contain b, and as such $\{b, \bar{b}\}$ is a complete family of conditions for $\mathcal{H}$. We can strengthen the conditions to obtain $\{bc, b\bar{c}, \bar{b}\}$. Minimal transversals of $\mathcal{H}$ that do not contain b or c necessarily contain a (as ab or ac would not be covered otherwise). Hence $\{bc, ab\bar{c}, a\bar{b}\}$ is a complete family of conditions for $\mathcal{H}$.

Let M be a maximum set of pairwise distinct 2-edges (matching) of $\mathcal{H}$. By removing k vertices we decrease the size of a maximum matching by at most k. Thus we have

$$\Delta(\mathcal{H}, \mathcal{H}_A) \geq \begin{cases} 2 - 2\alpha & \text{for } A \in \{\{bc\}, \{a\bar{b}\}\} \\ 3 - 3\alpha & \text{for } A \in \{\{ab\bar{c}\}\} \end{cases}. \tag{2.2}$$

Equation (2.1) becomes $2\tau_1^{2\alpha-2} + \tau_1^{3\alpha-3} = 1$. For our chosen α, we have that $\tau_1 \leq \tau_0$.

Case 2: $d_2(a) = 1$: the hypergraph $\mathcal{H}$ contains a vertex a from S that belongs to a unique 2-edge ab. We break down this case into two sub-cases depending on whether or not a belongs to some 3-edges: $d_3(a) = 0$ and $d_3(a) \geq 1$.

Since a is in a unique 2-edge, when one removes a and b, the size of a maximum matching decreases at most by 1.

- $d_3(a) = 0$: a is in a single 2-edge ab and no 3-edges. A minimal transversal of $\mathcal{H}$ either contains or does not contain b. As such, $\{b, \bar{b}\}$ is a complete family of conditions for $\mathcal{H}$. As ab is the only edge containing a, a minimal transversal of $\mathcal{H}$ that contains b cannot contain a. Similarly, every minimal transversal of $\mathcal{H}$ that does not contain b necessarily contains a. This makes $\{b\bar{a}, a\bar{b}\}$ a complete family of conditions for $\mathcal{H}$.

  Let M be a maximum set of pairwise disjoint 2-edges of $\mathcal{H}$. As ab is the only edge containing a, b belongs to one of the edges in M. The hypergraphs $\mathcal{H}_{b\bar{a}}$ and $\mathcal{H}_{a\bar{b}}$ contain all the edges in M except for the one containing b. Thus, $m(\mathcal{H}_{b\bar{a}}) = m(\mathcal{H}_{a\bar{b}}) \geq m(\mathcal{H}) - 1$. Since $|V(\mathcal{H}_{b\bar{a}})| = |V(\mathcal{H}_{a\bar{b}})| \leq |V(\mathcal{H})| - 2$, we have

  $$\Delta(\mathcal{H}, \mathcal{H}_A) \geq 2 - \alpha \text{ for } A \in \{\{b\bar{a}\}, \{a\bar{b}\}\}. \tag{2.3}$$

  Equation (2.1) becomes $2\tau_{2.1}^{\alpha-2} = 1$. For our chosen α, we have that $\tau_{2.1} \leq \tau_0$.

- $d_3(a) \geq 1$: a is in a single 2-edge and in some 3-edges, one of which being acd. We start with the conditions $\{bc, bd\bar{c}, b\overline{cd}, \bar{b}\}$. Any minimal transversal of $\mathcal{H}$ that does not contain either b or both c and d necessarily contains a. This makes $\{bc, bd\bar{c}, ab\bar{c}d, a\bar{b}\}$ a complete family of conditions for $\mathcal{H}$. We obtain

  $$\Delta(\mathcal{H}, \mathcal{H}_A) \geq \begin{cases} 2 - 2\alpha & \text{if } A = \{bc\} \\ 3 - 3\alpha & \text{if } A = \{bd\bar{c}\} \\ 4 - 3\alpha & \text{if } A = \{ab\overline{cd}\} \\ 2 - \alpha & \text{if } A = \{a\bar{b}\} \end{cases}. \tag{2.4}$$

Equation (2.1) becomes $\tau_{2.2}^{2\alpha-2} + \tau_{2.2}^{3\alpha-3} + \tau_{2.2}^{3\alpha-4} + \tau_{2.2}^{\alpha-2} = 1$. For our chosen α, we have that $\tau_{2.2} \leq \tau_0$.

Case 3: $d_2(a) = 0$ and $d_3(a) \geq 1$: the hypergraph $\mathcal{H}$ contains a vertex a from S that is in no 2-edge and in some 3-edges, one of which being abc. We start with the conditions $\{b, c\bar{b}, \overline{bc}\}$ and strengthen them to $\{b, c\bar{b}, a\overline{bc}\}$. Since we do not have any 2-edge anymore, we cannot decrease the size of a maximum matching. We obtain

$$\Delta(\mathcal{H}, \mathcal{H}_A) \geq \begin{cases} 1 & \text{if } A = \{b\} \\ 2 & \text{if } A = \{c\bar{b}\} \\ 3 & \text{if } A = \{a\overline{bc}\} \end{cases} . \tag{2.5}$$

Equation (2.1) becomes $\tau_3^{-1} + \tau_3^{-2} + \tau_3^{-3} = 1$. For our chosen α, we have that $\tau_3 \leq \tau_0$. □

This proof ensures that there is a measure μ and a descendant function ρ for our class of hypergraphs such that ρ is μ-bounded by 1.8393. This allows us to formulate the following theorem.

Theorem 18. *The number of minimal transversals in an hypergraph belonging to the class $\mathcal{C}$ is less than* $1.8393^{n-|S|}$.

Proof. Let μ and ρ be the measure and descendant function mentioned in Lemma 17. The height $h(\mathcal{T}_{\mathcal{H}})$ of the tree is bounded by $n - |S|$ so a straightforward application of Theorem 16 yields the result. □

Theorem 11 is a straightforward corollary of Theorem 18.

Theorem 11. *For any integer n,*

$$f_3(n) \leq 1.8393^{2n/3} \leq 1.5012^n.$$

Proof. The vertices of a tripartite 3-uniform hypergraph can be partitioned into three minimal transversals so any of them can be S. The minimization of the bound is achieved by using the biggest set which, in the worst case, has size n/3. □

Theorem 19. *Let $\mathcal{C}$ by a 3-context such that each dimension has size s. Then $\mathcal{C}$ has at most* $1.8393^{2s} \approx 3.3831^s$ *3-concepts.*

This theorem refers to cubic 3-contexts, *i.e.* when the all the dimensions have the same size. It is hard to generalise further the bound on the number of minimal transversals since 3-context induce a given partition of vertices that may not maximise the number of minimal transversals. It still gives good insight on the maximum number of 3-concepts, since the worst case context ought to have equilibrated dimensions.

2.3 Powerset d-Lattices and Multiplicative Construction

2.3.1 Powerset d-Lattice

In the 2-dimensional setting, what we call a powerset lattice on a ground set of size s is the Boolean lattice of all the subsets of $\mathcal{S} = \{1, \dots, s\}$, ordered by inclusion. It has 2^s elements. The corresponding 2-context is called a contranominal scale, and is the context $C = (\mathcal{S}, \mathcal{S}, \neq)$. In a Boolean concept lattice, all the parts of $\mathcal{S}$ are intents, and extents. A concept has the form $(S, \overline{S})$, with S a subset of $\mathcal{S}$.

In terms of hypergraph transversals, the corresponding hypergraph is a perfect matching (a set of disjoint edges), as shown in Figure 2.8.

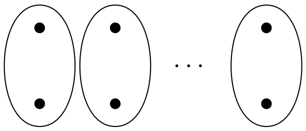


Figure 2.8: A contranominal scale gives a perfect matching. One can build 2^s minimal transversals by choosing an element out of each of the s edges.

The equivalent in 3-dimensions would be a triadic powerset, represented by the triadic contranominal scale $(\mathcal{S}, \mathcal{S}, \mathcal{S}, \mathcal{S}^3 \setminus \{(a, a, a) \mid a \in \mathcal{S}\})$ (Figure 2.9). The 3-lattice of all the 3-concepts induced by a triadic contranominal scale is called a powerset 3-lattice (see Figure 1.12). The 3-concepts of a powerset 3-lattice are of the form (S_1, S_2, S_3) where $S_1 \cap S_2 \cap S_3 = \emptyset$ and $S_i \cup S_j = \mathcal{S}$, for every combination of two dimensions.

	1	2	3	1	2	3	1	2	3
1		×	×	×	×	×	×	×	×
2	×	×	×	×		×	×	×	×
3	×	×	×	×	×	×	×	×	
		1			2			3	

Figure 2.9: The triadic contranominal scale on a ground set of size 3. This 3-context has twenty-seven 3-concepts.

Proposition 20. *A powerset 3-lattice on a ground set of size s has 3^s 3-concepts.*

Proof. The proof is straightforward by switching to minimal hypergraph transversals. Let $A = \{a_1, \dots, a_s\}$, $B = \{b_1, \dots, b_s\}$ and $C = \{c_1, \dots, c_s\}$ be sets. The hypergraph $\overline{\mathcal{H}} = (V, \mathcal{E})$ where $V = A \cup B \cup C$ and $\mathcal{E} = \bigcup_{i \in \mathcal{S}} \{a_i, b_i, c_i\}$ with $a_i \in A, b_i \in B, c_i \in C$ is exactly the complement hypergraph for the triadic contranominal scale.

All the 3-edges of $\overline{\mathcal{H}}$ are disconnected. $\overline{\mathcal{H}}$ has s 3-edges. For each minimal transversal of $\overline{\mathcal{H}}$, one can choose s times from 3 possibilities, hence 3^s concepts. □

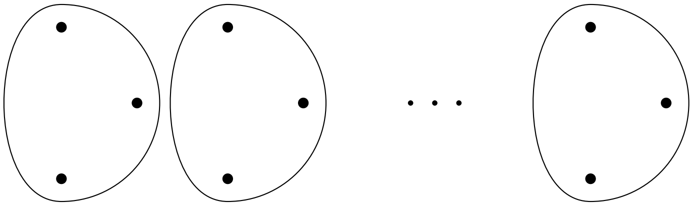


Figure 2.10: A triadic contranominal scale gives a set of disjoint 3-edges. One can build 3^s minimal transversals by choosing an element out of each of the s edges.

Figure 2.10 shows the complement hypergraph of a triadic contranominal scale. Clearly, this proposition can be extended to the d-dimensional case.

Proposition 21. *A powerset d-lattice on a ground set of size s has $d^s = 2^{s \log d}$ d-concepts.*

Proof. The proof is the same as before. We build a hypergraph $\overline{\mathcal{H}}$ that is the complement hypergraph of a d-adic contranominal scale. This hypergraph has s d-edges, hence d^s minimal transversals. □

2.3.2 Multiplicative construction

In [3, Chapter 5.2], Ganter and Wille present some gluings, some ways of putting together lattices by gluing them along some common substructures. They introduce the vertical and horizontal sums, and the union of contexts. In this section, we present another construction that allows to merge two d-contexts. This construction ensures that merging two d-contexts with respectively N and N' concepts, builds a d-context with $N \times N'$ d-concepts.

Let $\mathcal{C}^1 = (\mathcal{S}^1_1, \dots, S^1_d, \mathcal{R}^1)$ and $\mathcal{C}^2 = (\mathcal{S}^2_1, \dots, \mathcal{S}^2_d, \mathcal{R}^2)$ be two d-contexts such that the S^1_i, S^2_i are disjoint. The context $\mathcal{C} = (\mathcal{S}_1, \dots, \mathcal{S}_d, \mathcal{R})$ is constructed from the contexts $\mathcal{C}^1$ and $\mathcal{C}^2$ in the following way.

The dimensions of $\mathcal{C}$ are the union of the dimensions of C^1 and $\mathcal{C}^2$. The existing crosses are conserved. New crosses are added in all the cells that borrow some of their coordinates from different d-contexts. More formally,

$$\mathcal{S}_i = \mathcal{S}^1_i \cup \mathcal{S}^2_i, \forall i \in \{1, \dots, d\}$$

and

$$\mathcal{R} = \mathcal{R}^1 \cup \mathcal{R}^2 \cup \{(x_1, \dots, x_d) \mid \exists i, j \in \{1, \dots, d\} \text{ such that } x_i \in \mathcal{S}^1_i, x_j \in \mathcal{S}^2_j\}.$$

An example of this construction in two dimensions is shown in Figure 2.11, and in three dimensions in Figure 2.12.

This construction allows to extend any pair of concepts where one comes from each context into a new concept for the whole. This gives us the following proposition.

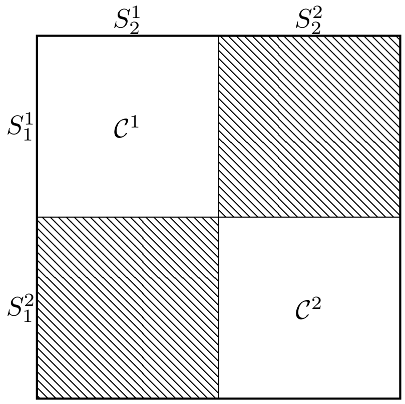


Figure 2.11: The aforementioned construction amounts to merging the two 2-contexts on a diagonal, while filling the rest with crosses. The grey areas represent this filling.

Proposition 22. *Let $\mathcal{C}$ be a context built from $\mathcal{C}^1$ and $\mathcal{C}^2$ by the aforementioned procedure. Let $X^1 = (X_1^1, \ldots, X_d^1)$ be a concept of $\mathcal{C}^1$, and $X^2 = (X_1^2, \ldots, X_d^2)$ be a concept of $\mathcal{C}^2$. Then $X = (X_1^1 \cup X_1^2, \ldots, X_d^1 \cup X_d^2)$ is a concept of $\mathcal{C}$.*

Proof. We have to prove that (A) $(X_1^1 \cup X_1^2, \ldots, X_d^1 \cup X_d^2)$ is indeed a box full of crosses in $\mathcal{R}$ and (B) this box is maximal. We set $X_i = X_i^1 \cup X_i^2$, for all $i \in \{1, \ldots, d\}$.

(A) We have to show that $\forall x_1 \in X_1, \ldots, x_d \in X_n, (x_1, \ldots, x_d) \in \mathcal{R}$. Either $x_1 \in \mathcal{S}_1^a, x_2 \in \mathcal{S}_2^a, \ldots, x_d \in \mathcal{S}_d^a$, and then we are either in $\mathcal{R}^a$, with a being 1 or 2, and the claim is true, or $\exists i, j \in \{1, \ldots, d\}$ such that $x_i \in \mathcal{S}_i^1, x_j \in \mathcal{S}_j^2$, so the crosses that are in $\mathcal{R}^1$ and $\mathcal{R}^2$ form a concept, since they can be extended "on the size".

(B) We have to show that for all $i \in \{1, \ldots, d\}$ there is no $z \in \mathcal{S}_i \setminus X_i$ such that $z \times \prod_{j \in \{1,\ldots,d\} \setminus i} X_j \subseteq \mathcal{R}$. Suppose that the concept can be extended on its first component, that is $\exists z \in \mathcal{S}_1$, such that $\{z\} \times \prod_{j \in \{2,\ldots,d\}} X_j \subseteq \mathcal{R}$. In particular, if $z \in \mathcal{S}_1^a$, this means that $\{z\} \times \prod_{j \in \{2,\ldots,d\}} X_1^a \subseteq \mathcal{R}^a$. That is a contradiction with the fact that $(X_1^1, \ldots, X_d^1)$ is a concept of $\mathcal{C}^a$. □

When stated in terms of hypergraph transversals in the complementary hypergraph of new context $\mathcal{C}$, Prop. 22 amounts to consider separately the minimal transversals of $\overline{\mathcal{H}}_{\mathcal{C}^1}$ and $\overline{\mathcal{H}}_{\mathcal{C}^2}$ and combining the solutions. The previous proposition is re-stated in these terms in the folowing proposition in a way that allows to glue k d-contexts.

Proposition 23. *Let k be an integer, and for each i between 1 and k, let $\mathcal{H}_i$ be a hypergraph of order n_i with t_i minimal transversals. Then the hypergraph $\mathcal{H}$ obtained by disjoint union of the k hypergraphs is of order $\sum_{i=1}^k n_i$ and has $\prod_{i=1}^k t_i$ minimal transversals. Moreover, if for every i, $\mathcal{H}_i$ is d-partite and d-uniform, then $\mathcal{H}$ is d-partite and d-uniform.*

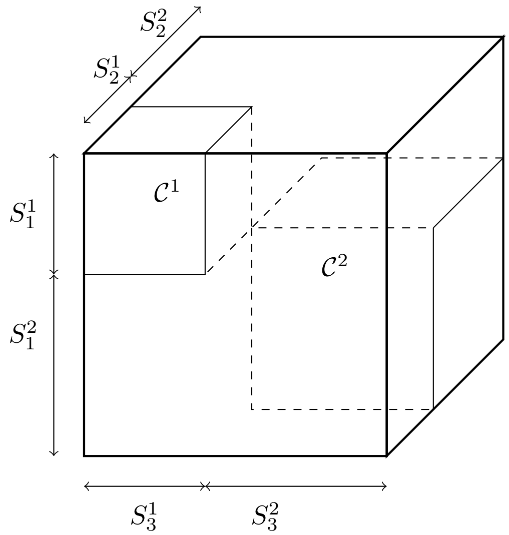


Figure 2.12: The two 3-contexts $\mathcal{C}^1$ and $\mathcal{C}^2$ are in a diagonal. Here, the crosses are not represented, but all that is not in C^1 or C^2 is full of crosses.

Proof. It is sufficient to look at each connected component of n_i vertices. The minimal transversals of the whole hypergraph are exactly the sets of vertices resulting from the union of one minimal transversal from each connected component, hence the number $\prod_{i=1}^{k} t_i$. Clearly, the disjoint union of the $\mathcal{H}_i$, for all i between 1 and k is still a d-partite d-uniform hypergraph. □

Corollary 24. $|\mathcal{T}(\mathcal{C})| = |\mathcal{T}(\mathcal{C}^1)| \times |\mathcal{T}(\mathcal{C}^2)|$.

Since any pair of concepts forms a new concept, the total number of concepts in the new context resulting from the construction is the product of the number of concepts in each original context. Moreover, the sizes of the d-contexts are only summed.

Remark 25. *The 3-context $(a, a, a, \emptyset)$, where a is a dimension consisting of only one element and the relation is empty, has 3 concepts: $(a, a, \emptyset)$, $(a, \emptyset, a)$ and $(\emptyset, a, a)$. When applying the construction mentioned above s times iteratively on this empty context with itself, one builds a context with*

$$\underbrace{3 \times 3 \times \cdots \times 3}_{s \text{ times}} = 3^s$$

3-concepts, that is exactly the triadic contranominal scale. When doing the same in d-dimensions, one builds a d-adic contranominal scale.

Figure 2.13 shows another angle of the multiplicative construction that gives a triadic contranominal scale.

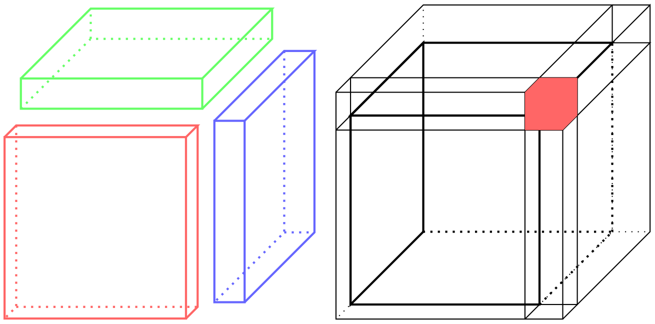


Figure 2.13: By adding new rows, layers and “columns” full of crosses on the side of the 3-context and letting only the red cell be empty, the number of concepts is multiplied by three. Such construction gives a triadic contranominal scale.

2.3.3 A Lower Bound on the Maximal Size of a 3-Lattice

For the sake of completeness, as we gave an upper bound on $f_3(n)$ it would be interesting to have some attainable numbers. The construction of disjoint 3-edges shown in Figure 2.10 gives $3^s = 3^{n/3} \approx 1.4422^n$ minimal transversals. In this section, we prove that $f_3(n) \geq c1.4977^n$.

To prove this theorem, we use the multiplicative construction from the previous section to build some contexts with a high number of concepts. A computer search on small instances allows us to formulate the following claim.

Claim 26. *There is a tripartite 3-uniform hypergraph on fifteen vertices with four hundred and twenty-eight minimal transversals:*

$$f_3(15) \geq 428.$$

This hypergraph is shown in Figure 2.14. We denote this hypergraph by $\mathcal{H}_{15}$.

	a	b	c	d	e	a	b	c	d	e	a	b	c	d	e	a	b	c	d	e	a	b	c	d	e
1	×						×						×						×						×
2		×						×						×						×	×				
3			×						×						×	×						×			
4				×						×	×						×						×		
5					×	×						×						×						×	
			α					β					γ					δ					ϵ		

Figure 2.14: Each cross represents a 3-edge. This hypergraph has four hundred and twenty-eight minimal transversals.

Theorem 27. *There exists a constant c such that for any integer n, $f_3(n) \geq c1.4977^n$.*

Proof. Let $c = (\frac{1.4422}{1.4977})^{12}$. We observe that $428 > 1.4977^{15}$. Now fix some integer n that is a multiple of 3. There are two integers k and r such that $n = 3(5k + r)$ with r between 0 and 4. By making k disjoint copies of $\mathcal{H}_{15}$ and then adding r disjoint edges of arity 3, we obtain a hypergraph with $428^k \times 3^r$ minimal transversals. Since

$428^k \geq 1.4977^{15k}$ and $3^r \geq 1.4422^{3r}$, the number of minimal transversals is more than $1.4977^{n-3r} \times 3^{3r}$. Since r is between 0 and 4, $(\frac{1.4422}{1.4977})^{3r}$ is at least c. □

Our extremal example $\mathcal{H}_{15}$, in terms of polyadic concept analysis, has this representation (Figure 2.15): It has exactly one "hole" per row, column and layer and is, as such, a solution to the 3-dimensional rook problem. It is in fact how we found it, and the computer search only confirmed that it was a good intuition. The computer search was not exhaustive and solutions with more 3-concepts might exist, even in $5 \times 5 \times 5$ contexts.

	a	b	c	d	e	a	b	c	d	e	a	b	c	d	e	a	b	c	d	e	a	b	c	d	e
1	▨	×	×	×	×	×	×	×	×	▨	×	×	×	▨	×	×	×	▨	×	×	×	▨	×	×	×
2	×	▨	×	×	×	▨	×	×	×	×	×	×	×	×	▨	×	×	×	▨	×	×	×	▨	×	×
3	×	×	▨	×	×	×	▨	×	×	×	▨	×	×	×	×	×	×	×	×	▨	×	×	×	▨	×
4	×	×	×	▨	×	×	×	▨	×	×	×	▨	×	×	×	▨	×	×	×	×	×	×	×	×	▨
5	×	×	×	×	▨	×	×	×	▨	×	×	×	▨	×	×	×	▨	×	×	×	▨	×	×	×	×
	α					β					γ					δ					ϵ				

Figure 2.15: This is a $5 \times 5 \times 5$ 3-context. It has 428 3-concepts.

Conclusion

In this chapter, we studied some aspects of extremal combinatorics of d-lattices. We chose a change of formalism in order to study our favorite objects under the scope of hypergraphs. This allowed us to look for minimal transversals.

Some considerations of combinatorics on d-contexts become easier to see when ported to transversals. It is the case for Theorem 27 for example. The multiplicative construction that led to this theorem also admits a simpler proof (and construction) when stated in terms of hypergraph transversals.

Although the proof for Theorem 11 is not that simple (it requires a small case study and some analytics results from the litterature), the tools that allowed us to reach that conclusion are tools from graph and hypergraph theory. Those tools are also widely applied in enumeration, so we finally came full circle to some known problems of Formal Concept Analysis.

The innocent question of "how many maximal boxes of dimension 3 can I fit into a 3-dimensional context" was not that innocent after all. Its study led us to some interesting proof technique (measure and conquer), and we reformulated this question under various formalisms that underline the links between data science and more theoretical aspects of discrete mathematics.

As it has been pointed out to me in July 2018 in Lyon by the mighty Stephan Thomassé, it is almost a shame to present a result where you give an interval for a value between 1.497^n and 1.501^n, without committing to the end so here it is.

Conjecture 28. $f_3(n) = \left(\frac{3}{2}\right)^n$

Chapter 3

The average case: what to expect?

With lattices and d-lattices, the worst case, in terms of size, is exponential (in the size of the dimensions). In the previous chapter, we investigated the worst case scenario in 3-lattices. In this chapter, we choose another approach, and consider the average case. The first part of the chapter is dedicated to implication bases as another type of information contained in the context. We study the average size of a particular implication base, under two different statistical models.

The second part of the chapter deals with the average number of concepts. First, we deal with the average number of 2-concepts under the same two models as for the implications. Then, we extend those results to the average number of d-dimensional concepts.

3.1 Implicational Bases

Due to the fact that they are so widely used in practice, implications (association rules that have a confidence of 100%) are one of the most studied notions in Formal

Concept Analysis and data mining in general. An example of association rule can be stated as follow: “men who buy diapers tend to buy beer”. The confidence of the rule is the percentage of men who do buy beer as a reward for themselves when they buy diapers. Implications have a stronger meaning, since it would be that “all men who buy diaper buy beer”, a rule with confidence 100%.

In terms of Formal Concept Analysis, implications are defined as follows.

Definition 29. *An* implication *(between attributes) is a pair of sets $X, Y \subseteq \mathcal{A}$. It is noted $X \to Y$. An implication $X \to Y$ is said to* hold *in a context $\mathcal{C}$ if and only if $X' \subseteq Y'$.*

In particular, the implication $X \to X$ always holds in a context. All the implications of the form $X \to Y$ when $Y \subseteq X$ hold too. For these reasons, many implications are redundant. The number of implications that hold can be quite large [17]. It is necessary to focus on the interesting ones.

Definition 30. *A set of implications that allows for the derivation of all the implications that hold in a context, and only them, through the application of Armstrong's axioms is called a* base*.*

We recall the Armstrong's axioms [18]: for any sets of attributes X, Y and Z, if $Y \subseteq X$ then $X \to Y$, $X \to Y$ then $X \cup Z \to Y \cup Z$ and if $X \to Y$ and $Y \to Z$, then $X \to Z$.

The closure system induced by the implicational system given by a context (called logical closure) is isomorphic to the one given by the concepts of this context.

3.1.1 Some Interesting Bases

One could ask himself, what is the smallest (cardinality wise) set of implications that forms a base. The answer to this question was given by Guigues and Duquenne in [19].

Definition 31 (Duquenne-Guigues Base)**.** *An attribute set P is a pseudo-intent if and only if $P \neq P''$ and $Q'' \subset P$ for every pseudo-intent $Q \subset P$. The set of all the implications $P \to P''$ in which P is a pseudo-intent is called the Duquenne-Guigues Base.*

The Duquenne-Guigues Base, also called *canonical* base, or *stem* base has first been introduced in [19] and is the smallest of all the bases. Here, we denote this base as Σ_{stem}. The study of the complexity of enumeration of this base started in [17] where it was shown that the size of the Duquenne-Guigues base can be exponential in the size of the input, and that computing the size of the stem base is a #-P-hard problem. Then, the problem of deciding whether a set of attributes is a pseudo-intent was shown to be in Co-NP in [20]. It was then shown to be CoNP-complete in [21]. Then, the problem of enumerating pseudo-intents was shown minimal-transversals-enumeration-hard in [22]. In this papier, it was also shown that the enumeration of pseudo-intents is impossible with total polynomial time if $P \neq NP$. Finally, the impossibility of enumerating pseudo-intents in reverse lexicographic order if $P \neq NP$ was shown in [23].

	a_1	a_2	a_3	a_4	a_5
o_1	×	×			
o_2		×		×	×
o_3		×	×	×	
o_4			×		×
o_5				×	×

Table 3.1: Toy context $\mathcal{C}$.

While the Duquenne-Guigues Base is the smallest base altogether, the *base of proper premises*, or *Canonical Direct Base*, noted here Σ_{Proper}, is the smallest base for which the logical closure can be computed with a single pass. The Canonical Direct Base was initially known under five independent definitions, shown to be equivalent by Bertet and Monjardet in [24].

Proposition 32 (from [3])**.** *$P \subseteq \mathcal{A}$ is a premise of $a \in \mathcal{A}$ if and only if $(\mathcal{A} \backslash o') \cap P \neq \emptyset$ holds for all $o \in \mathcal{O}$ such that $(o, a) \notin \mathcal{R}$. P is a proper premise for a if and only if P is minimal with respect to subset inclusion for this property.*

Proposition 23 from [3] uses $o \swarrow a$ instead of $(o, a) \notin \mathcal{R}$. It is a stronger condition that involves a maximality condition that is not necessary here.

The set of proper premises of an attribute is equivalent to the minimal transversals of a hypergraph induced from the context with the following proposition:

Proposition 33 (From [25])**.** *P is a premise of a if and only if P is a hypergraph transversal of $\mathscr{H}_a$ where*

$$\mathscr{H}_a = \{\mathcal{A} \setminus o' | o \in \mathcal{O}, (o, a) \notin \mathcal{R}\}$$

The set of all proper premises of a is exactly the transversal hypergraph $Tr(\mathscr{H}_a)$.

To illustrate this link, we show the computation of the proper premises for some attributes of context 3.1. We compute the hypergraph $\mathscr{H}_a$ for a_1, a_2 and a_5. Let us begin with attribute a_1. We have to compute $\mathscr{H}_{a_1} = \{\mathcal{A} \setminus o' \; | o \in \mathcal{O}, (o, a_1) \notin \mathcal{R}\}$ and $Tr(\mathscr{H}_{a_1})$. In $\mathcal{C}$, there is no cross for a_1 in the rows o_2, o_3, o_4 and o_5. We have:

$$\mathscr{H}_{a_1} = \{\{a_1, a_3\}, \{a_1, a_5\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_4\}\}$$

and

$$Tr(\mathscr{H}_{a_1}) = \{\{a_1\}, \{a_2, a_3, a_5\}, \{a_3, a_4, a_5\}$$

We have the premises for a_1, which give the two implications $a_2a_3a_5 \to a_1$ and $a_3a_4a_5 \to a_1$. $\{a_1\}$ is also a transversal of $\mathscr{H}_{a_1}$ but can be omitted here, since $a \to a$ is always true.

In the same way, we compute the hypergraph and its transversal hypergraph for the other attributes. For example,

$$\mathscr{H}_{a_2} = \{\{a_1, a_2, a_3\}, \{a_1, a_2, a_4\}\} \text{ and } Tr(\mathscr{H}_{a_2}) = \{\{a_1\}, \{a_2\}, \{a_3, a_4\}\}$$

$$\mathscr{H}_{a_5} = \{\{a_1, a_5\}, \{a_3, a_4, a_5\}\} \text{ and } Tr(\mathscr{H}_{a_5}) = \{\{a_5\}, \{a_1, a_3\}, \{a_1, a_4\}\}$$

The set of all proper premises of a_i is exactly the transversal hypergraph $Tr(\mathscr{H}_{a_i})$, $\forall i \in \{1, \dots, 5\}$, to which we remove the trivial transversals (a_i is always a transversal for $\mathscr{H}_{a_i}$). The base of proper premises for context $\mathcal{C}$ is the union of the proper premises for each attribute:

$$\Sigma_{Proper}(\mathcal{C}) = \bigcup_{a \in \mathcal{A}} Tr(\mathscr{H}_a) \setminus \{a\}$$

It should be noted that the complexity of the enumeration problem was shown to be quasi-polynomial in the seminal paper by Fredman and Khachyian [26].

3.2 Average Size of Implication Bases

In [25], Ryssel, Distel and Borchmann provided expected numbers of proper premises and concept intents. Their approach, like the one in [27], uses the Erdős-Rényi model [28] to generate random hypergraphs. However, in [25], the probability for each vertex to appear in a hyperedge is a fixed 0.5 (by definition of the model) whereas the approach presented in [27] consider this probability as a variable of the problem and is thus more general.

3.2.1 Single parameter model

In the following, we assume all sets to be finite, and that $|\mathcal{O}|$ is polynomial in $|\mathcal{A}|$. We call p the probability that an object o has an attribute a. An object having an attribute is independent from other attributes and objects. We denote by $q = 1 - p$ the probability that $(o, a) \notin \mathcal{R}$. The probability of an attribute that is not a appearing in a hyperedge of $\mathscr{H}_a$ is also q.

The hypergraphs that we consider in the following are sub-hypergraphs constructed from $\mathscr{H}_a$ by removing a and removing all the hyperedges that contained only a. The transversal hypergraph of a hypergraph constructed in this way is exactly $Tr(\mathscr{H}_a) \setminus \{a\}$. This allows us to consider the transversal hypergraph without adding a as a premise for a. The average number of hyperedges of this hypergraph is $m = |\mathcal{O}| \times q \times (1 - p^{|\mathcal{A}|-1})$. Indeed, there is one hyperedge for each object o for which $(o, a) \notin \mathcal{R}$ and there exists an attribute a_2 such that $(o, a_2) \notin \mathcal{R}$ (otherwise the edge would be empty and, as such, removed). We note n the number of vertices of $\mathscr{H}_a \setminus \{a\}$. At most all attributes appear in $\mathscr{H}_a \setminus \{a\}$, except a, so $n \leq |\mathcal{A}| - 1$.

Proposition 34 (Reformulated from [27])**.** *In a random hypergraph with m edges and n vertices, with $m = \beta n^\alpha, \beta > 0$ and $\alpha > 0$ and a probability p that a vertex appears in an edge, there exists a positive constant c such that the average number of minimal transversals is*

$$O\left(n^{d(\alpha) log_{\frac{1}{q}} m + c \ln \ln m}\right)$$

with $q = 1 - p$, $d(\alpha) = 1$ if $\alpha \leq 1$ and $d(\alpha) = \frac{(\alpha+1)^2}{4\alpha}$ otherwise.

Proposition 34 bounds the average number of minimal transversals in a hypergraph where the number of edges is polynomial in the number of vertices. In [27], the authors also prove that this quantity is quasi-polynomial.

From Proposition 34 we straightforwardly deduce the following property for the number of proper premises for an attribute.

Proposition 35. *In a random context with $|\mathcal{A}|$ attributes, $|\mathcal{O}|$ objects and probability p that $(o, a) \in \mathcal{R}$, the number of proper premises for an attribute is on average:*

$$O\left((|\mathcal{A}|-1)^{\left(d(\alpha)log_{\frac{1}{p}}\left(|\mathcal{O}|\times(q\times(1-p^{|\mathcal{A}|-1}))\right)+c\ln\ln\left(|\mathcal{O}|\times(q\times(1-p^{|\mathcal{A}|-1}))\right)\right)}\right)$$

and is quasi-polynomial in the number of objects.

Proposition 35 states that the number of proper premises of an attribute is on average quasi-polynomial in the number of objects in a context where the number of objects is polynomial in the number of attributes.

As attributes can share proper premises, $|\Sigma_{Proper}|$ is on average less than

$$|\mathcal{A}| \times O\left((|\mathcal{A}|-1)^{\left(d(\alpha)log_{\frac{1}{p}}\left(|\mathcal{O}|\times q\times(1-p^{|\mathcal{A}|-1}))\right)+c\ln\ln\left(|\mathcal{O}|\times q\times(1-p^{|\mathcal{A}|-1}))\right)\right)}\right).$$

Since $|\Sigma_{stem}| \leq |\Sigma_{Proper}|$, Proposition 35 immediately yields the following corollary:

Corollary 36. *The average number of pseudo-intents in a context where the number of objects is polynomial in the number of attributes is less than or equal to:*

$$|\mathcal{A}| \times O\left((|\mathcal{A}|-1)^{\left(d(\alpha)log_{\frac{1}{p}}\left(|\mathcal{O}|\times q\times(1-p^{|\mathcal{A}|-1})\right)+c\ln\ln\left(|\mathcal{O}|\times q\times(1-p^{|\mathcal{A}|-1})\right)\right)}\right)$$

Corollary 36 states that in a context where the number of object is polynomial in the number of attributes, the number of pseudo-intents is on average at most quasi-polynomial.

3.2.2 Almost sure lower bound for the number of proper premises

An almost sure lower bound is a bound that is true with probability close to 1. In [27], the authors give an almost sure lower bound for the number of minimal transversals.

Proposition 37 (Reformulated from [27])**.** *In a random hypergraph with m edges and n vertices, and a probability p that a vertex appears in an edge, the number of minimal transversals is almost surely greater than*

$$\mathcal{L}_{MT} = n^{log_{\frac{1}{q}} m + O(\ln\ln m)}$$

Proposition 37 states that in a random context with probability p that a given object has a given attribute, one can expect at least $\mathcal{L}_{MT}$ proper premises for each attribute.

Proposition 38. *In a random context with $|\mathcal{A}|$ attributes, $|\mathcal{O}|$ objects and probability q that a couple $(o, a) \notin \mathcal{R}$, the size of Σ_{Proper} is almost surely greater than*

$$|\mathcal{A}| \times (|\mathcal{A}| - 1)^{\left(log_{\frac{1}{p}}\left(|\mathcal{O}| \times q \times (1 - p^{|\mathcal{A}|-1})\right) + O\left(\ln \ln\left(|\mathcal{O}| \times q \times (1 - p^{|\mathcal{A}|-1})\right)\right)\right)}$$

As Prop 38 states a lower bound on the number of proper premises, no bound on the number of pseudo-intents can be obtained this way.

3.2.3 Multiparametric model

In this section we consider a multiparametric model that fits real life data better. In this model, each attribute j has a probability p_j of appearing in the description of a given object. All the attributes are not equiprobable.

We consider a context with m objects and n attributes. The set of attributes is partitioned into 3 subsets:

- The set U contains the attributes that appear in a lot of objects' descriptions (ubiquitous attributes). For all attributes $u \in U$ we have $q_u = 1 - p_u < \frac{x}{m}$ with x a fixed constant.

- The set R represents rare events, i.e. attributes that rarely appear. For all attributes $r \in R$, we have that $p_r = 1 - e^{-\frac{1}{\ln n}}$ tends to 0.

- The set $F = \mathcal{A} \setminus (U \cup R)$ of other attributes.

Proposition 39 (Reformulated from theorem 3 [27])**.** *In the multiparametric model, we have:*

- *If $|F \cup R| = O(\ln |\mathcal{A}|)$, then the size of the base of proper premises is on average at most polynomial.*

- *If $|R| = O((\ln |\mathcal{A}|)^c)$, then the size of the base of proper premises is on average at most quasi-polynomial.*

- *If $|R| = \Theta(|\mathcal{A}|)$, then the size of the base of proper premises is on average at most exponential on $|R|$.*[1]

Proposition 39 states that when most of the attributes are common (that is, are in the set U), $|\Sigma_{Proper}|$ is on average at most polynomial. When there is a logarithmic quantity of rare attributes (attributes in R), $|\Sigma_{Proper}|$ is on average at most quasi-polynomial (in the number of objects). When most of the attributes are rare events, $|\Sigma_{Proper}|$ is on average at most exponential.

As in the single parameter model, Proposition 39 also yields the same bounds on the number of pseudo-intents.

[1] We say that a function is in $\Theta(n)$ if, when n gets large enough, the function is at least $k_1 \times n$ and at most $k_2 \times n$ for some constants k_1 and k_2.

3.3 d-Dimensional Implications

Until recently (as of 2018), implications in d-contexts were not formally defined. In [29], Ganter and Obiedkov discuss the matter of implications in triadic contexts, however, the paper does not contain a formal proof that the set of implication they introduce allows to build the corresponding 3-lattice. In [30], Bazin introduced an implicational base that allows to build the corresponding d-lattice. It is this implicational base that we study here.

In this section, we introduce some notations necessary to understand this d-dimensional implication base and explain its computation. From this, we estimate the average number of d-implications.

Definition 40. *Let k be a dimension of a d-context $(\mathcal{S}_1, \ldots, \mathcal{S}_d, \mathcal{R})$. Let $\overline{k} = \{1, \ldots, d\} \setminus \{k\}$ be the set of all the other dimensions. Then $\mathcal{C}^{(k,\overline{k})}$ is the 2-context $(\mathcal{S}_k, \prod_{i\in\overline{k}} \mathcal{S}_i, \mathcal{R}^k)$ where $\mathcal{R}^k = \{(s_k, s_2) \mid (s_k, \prod_{i\in\overline{k}} s_i) \in \mathcal{R}\}$.*

	a	b	c	a	b	c	a	b	c
α	×	×		×			×		
β	×			×			×	×	
γ	×			×		×		×	×
		1			2			3	

	(a,1)	(a,2)	(a,3)	(b,1)	(b,2)	(b,3)	(c,1)	(c,2)	(c,3)
α	×	×	×	×					
β	×	×	×			×			
γ	×	×				×		×	×

Figure 3.1: An example of a 3-context $\mathcal{C} = (Numbers, Greek, Latin, \mathcal{R})$ and $\mathcal{C}^{(Greek,\{Latin,Numbers\})}$.

Figure 3.1 shows an example of a 3-context and a 2-context constructed by fixing a dimension and applying a Cartesian product to the others. This illustrates the previous definition.

Theorem 41 ([30, Theorem 1])**.** *Let $\mathcal{C} = (\mathcal{S}_1, \ldots, \mathcal{S}_d, \mathcal{R})$ be a d-context and k a dimension. An implication base of $\mathcal{C}^{(k,\overline{k})}$ is an implication base of $\mathcal{C}$.*

Theorem 41 states that from the implications of $\mathcal{C}^{(k,\overline{k})}$, one can derive a structure that is isomorphic to the d-lattice of d-context $\mathcal{C}$. Since any 2-dimensional base works, the base of Proper Premises gives an upper bound on the minimal number of d-implications. The Duquenne-Guigues base, while still the smallest base for context $\mathcal{C}^{(k,\overline{k})}$ might not be the smallest for $\mathcal{C}$ altogether.

3.3.1 Single Parameter Model

In this case, the model is straightforwardly the same as in Section 3.2.1. For each cell in the d-dimensional context, there is a probability p that there is a cross. We denote by $q = 1 - p$ the probability that this cell is empty.

For a given attribute a of the form $(s_2, \ldots, s_d)$ (remember that the attributes are elements of the Cartesian product on $d-1$ dimensions), we study $\mathscr{H}_a$. The probability that another attribute appears in an edge of $\mathscr{H}_a$ is q. The comparison with the 2-dimensional case stops here, as the number of vertices and edges is radically different in this case.

The hypergraph $\mathscr{H}_a$ has a vertex for every element of the Cartesian product of the dimensions. Let k be the dimension that serves as pivot in $\mathcal{C}^k$. Then $\mathscr{H}_a$ has at most $n = \prod_{i \in \{1,\ldots,d\} \setminus \{k\}} |\mathcal{S}_i|$ vertices. The hypergraph $\mathscr{H}_a$ has an edge for every element of the pivot dimension k that does not have a but has at least another element. The average number of edges is then $m = |\mathcal{S}_k| \times q \times (1 - p^n)$.

The number of vertices of this hypergraph is big ($\prod_{i \in \{1,\ldots,d\} \setminus \{k\}} |\mathcal{S}_i|$ where k is the dimension chosen as pivot) while its number of edges is small. The average number of implications in the d-dimensional implication base defined in [30] is then the average number of minimal transversals given by Proposition 37, using the number of vertices and the number of edges that we calculated.

It is not clear what the multiparameter model would be like in d dimensions. For this reason, we do not include an analysis of the average number of implications in this case.

3.4 Average Number of concepts

Given a context, counting the number of concepts is as hard as enumerating them, as it has been shown by Kuznetsov in [31]. The average number of concepts has been studied more than the average number of implications. Some papers exist on the 2-dimensional case: in 2005 by Lhote, Rioult and Soulet [32, 33], in 2009 by Emilion and Lévy [34] for the probability of a concept, by Klimushkin, Obiedkov and Roth [35] and more recently by Bodini, David and Bazin, in a paper not yet published [36].

3.4.1 2-Dimensions

This results comes from [36]. We consider a context with n objects and m attributes, such that n and m are polynomially related, that follows the same single parameter model described above (each cell has a probability p of having a cross, independently from any other cell).

Let us start by computing the probability that a pair (X_1, X_2) such that X_1 is a set of objects of size x and X_2 is a set of attributes of size y is a concept. We want $X_1 \times X_2$ to be full of crosses. The probability that this happens is p^{xy}. For the maximality of (X_1, X_2) to be true, we want a 0 in each line and each column outside the concept (Figure 3.2). For the columns, we obtain the probability $(1 - p^y)^{n-x}$ and for the lines $(1 - p^x)^{m-y}$. All in all, we obtain that the probability that a pair objects-attributes (X_1, X_2) is a concept is:

$$\mathtt{P}((X_1, X_2) \text{ is a concept}) = p^{xy} \times (1 - p^y)^{n-x} \times (1 - p^x)^{m-y}.$$

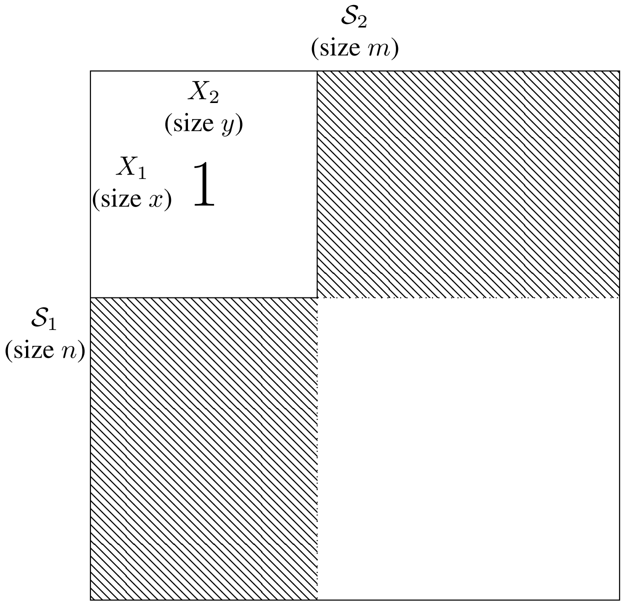


Figure 3.2: For (X_1, X_2) to be a concept, the area formed by $X_1 \times X_2$ has to be full of ones. Moreover, the grey areas have to have at least one zero per line (resp. column) to ensure the maximality of the concept. What happens in the white area in the South-East is irrelevant.

Summing on the possible sizes for X_1 and X_2, we obtain that the average number of concepts of a context $\mathcal{C}$ that has n objects and m attributes and follows the single parameter model follows the following formula:

$$\mathbb{E}(\mathcal{C}) = \sum_{x=1}^{n} \left(\binom{n}{x} \sum_{y=1}^{m} \binom{m}{y} p^{xy} \times (1-p^y)^{n-x} \times (1-p^x)^{m-y} \right).$$

After rewriting the expression, the authors of [36] conclude that this quantity is quasi-polynomial.

In the following, we extend this result to the multiparameter models. We are interested in a context $\mathcal{C} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{R})$ where $\mathcal{S}_1$ has size n, $\mathcal{S}_2$ has size m and for each element a of $\mathcal{S}_2$, we have p_a the probability that there is a cross in each cell of this column. We start by computing the probability that a pair of sets (X_1, X_2) where X_1 comes from $\mathcal{S}_1$ and X_2 from $\mathcal{S}_2$ is a concept. We denote the sizes of X_1 and X_2 by x and y, respectively.

We want $X_1 \times X_2$ to be full of crosses. The probability that this happens is $\prod_{\ell \in X_2} p_\ell^x$. For the maximality of (X_1, X_2) to be true, we want a 0 in each line and each column outside the concept. For the columns, we obtain the probability $\prod_{a \in \mathcal{S}_2 \setminus X_2} (1 - p_a^x)$ and for the lines $(1 - \prod_{\ell \in X_2} p_\ell)^{n-x}$. All in all, we obtain that the probability that a pair objects-attributes (X_1, X_2) is a concept is:

$$\mathtt{P}((X_1, X_2) \text{ is a concept}) = \prod_{\ell \in X_2} p_\ell^x \times (1 - \prod_{\ell \in X_2} p_\ell)^{n-x} \times \prod_{a \in \mathcal{S}_2 \setminus X_2} (1 - p_a^x).$$

Summing on all the possibilities for X_1 and X_2, we obtain that the average number of concepts of a context $\mathcal{C}$ that has a set of object $\mathcal{S}_1$ of size n and a set of attributes $\mathcal{S}_2$ of size m, and follows the multi parameter models follows the following formula:

$$\sum_{X_1 \in 2^{\mathcal{S}_1}} \left(\sum_{X_2 \in 2^{\mathcal{S}_2}} \left(\prod_{\ell \in X_2} p_\ell^x \times (1 - \prod_{\ell \in X_2} p_\ell)^{n-x} \times \prod_{a \in \mathcal{S}_2 \setminus X_2} (1 - p_a^x) \right) \right).$$

3.4.2 d-Dimensions

The subject of the average number of d-concepts in a d-context has not been studied before. Here, we take on the single parameter model in order to use the same approach as before on a d-dimensional setting.

We consider a d-context $(\mathcal{S}_1, \ldots, \mathcal{S}_d, \mathcal{R})$ where the dimensions have s_i elements, for i in $\{1, \ldots, d\}$. Each cell has a probability p of having a cross. We start by computing the probability that a tuple of d sets $(X_1, \ldots, X_d)$, where each X_i comes from $\mathcal{S}_i$, and each set has a size x_i, is a concept.

We want $X_1 \times \cdots \times X_d$ to be full of crosses. The associated probability is $p^{\prod_{i=1}^d x_i}$. Then, for each dimension, we want to have at least one 0 in the "shadow" of our concept (see Figure 3.3). For this, we find $\prod_{i=1}^d (1 - p^{\prod_{j=1, j \neq i}^d x_j})^{s_i - x_i}$. Combining all this gives us

$$\mathtt{P}((X_1, \ldots, X_d) \text{ is a concept }) = p^{\prod_{i=1}^d x_i} \times \prod_{i=1}^{d} (1 - p^{\prod_{j=1, j \neq i}^d} X_j)^{s_i - x_i}$$

for the probability that a d-tuple is a d-concept. To ease notation in the following part, we will denote this probability $\mathtt{P}((x_1, \ldots, x_d))$.

Summing on all the possibilities for the X_i, we obtain

$$\sum_{x_1=1}^{s_1} \left(\binom{s_1}{x_1} \sum_{x_2=1}^{s_2} \left(\binom{s_2}{x_2} \cdots \sum_{x_d=1}^{s_d} \left(\binom{s_d}{x_d} \left(\mathtt{P}\left((x_1, \ldots, x_d) \right) \right) \right) \cdots \right) \right)$$

for the average number of concepts of a d-context $(\mathcal{S}_1, \ldots, \mathcal{S}_d, \mathcal{R})$ where the dimensions have size s_i and that follows the single parameter model.

Discussion and conclusion

The topic of randomly generated contexts is a central issue in Formal Concept Analysis, most notably when they are used to compare algorithm performances. Since 2002, where a paper by Kuznetsov and Obiedkov [37] compares some concept lattice

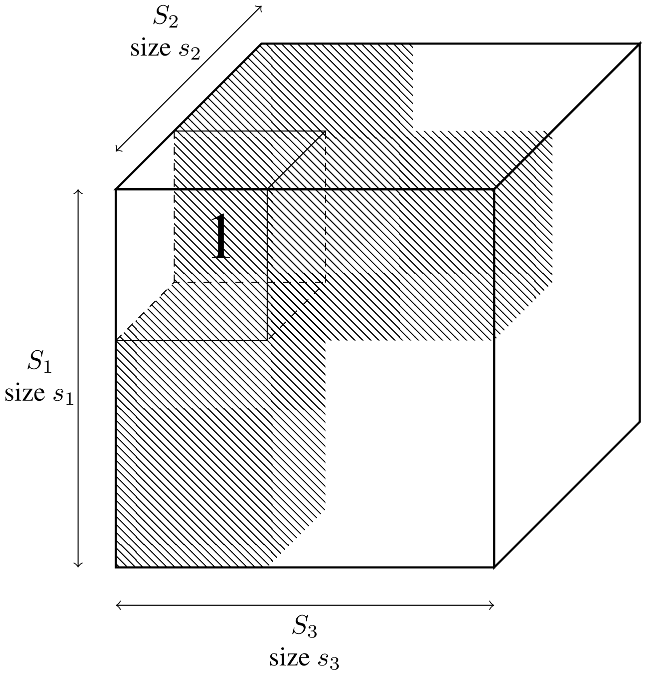


Figure 3.3: For (X_1, X_2, X_3) to be a concept, the area formed by $X_1 \times X_2 \times X_3$ has to be full of ones. Moreover, the grey area have to have at least one zero per line (resp. column or layer) to ensure the maximality of the concept. What happens in the white area in the rest of the cube is irrelevant.

building algorithms theoretically and experimentally, few experimental studies have been made. In [38], Borchmann and Hanika investigate the Stegosaurus phenomenon that arise when generating random contexts, where the number of pseudo-intents is correlated with the number of concepts.

In order to remove the problems raised by experiments on random contexts, in [39] Ganter discusses how to randomly and uniformly generate closure systems on up to seven elements.

In [40], the authors introduce a tool to generate less biased random contexts, avoiding repetition while maintaining a given density (inside an interval defined with some user-fixed lower and upper bounds), for any number of elements. This tools does not, however, ensure uniformity.

The multiparameter model that we used in this chapter allows for random contexts that are more similar to real life datasets [27], in the sense that it unties attributes from one another. We do not have any theoretical proof whether the multiparameter model avoids the Stegosaurus phenomenon. This issue is worth more investigation, both theoretical and experimental.

Part II

Algorithms

While the first part of this dissertation revolves around combinatorial problems on d-lattices, the second is dedicated to more algorithmic matters. We are not interested anymore in the cardinality of our structures, but rather in properties about those that allow to perform some operations on them.

In Chapter 4, we present an algorithm that enumerate the d-concepts of a d-context. As there is an important link between d-lattices and data mining problems, enumerating the elements of a d-lattice is an interesting problem. Due to the combinatorics of d-lattices, presented in the first part of the dissertation, we know that in the worst case the number of d-concepts is a huge drawback in applications. As a step to tackle this drawback, we developped an incremental algorithm. Such an algorithm starts from an incomplete solution and computes the complete solution. Our algorithm starts with a set of d-concepts from a d-context, and consider that the d-context is augmented with a new element on a dimension. As the saying goes, “when you’re tired of counting, you can always enumerate”.[2]

As enumerating all the d-concepts is still a hard task, we propose to shrink the structure. Chapter 5 shows another approach to takle the limitations induced by the size of d-lattices. Introducer concepts are the smallest concepts (with respect to a given order) that have a given element. They are well known and studied in the 2-dimensional case, as AOC-posets or Galois Sub-herarchies. They find a number of applications, most notably in software engineering (they were first mentioned in this context, and are still used, in particular in Chapter 6), and data mining. We define their equivalent in d-dimensions, the introducer d-concepts, and characterise their structure as a d-ordered set. We also generalise the notion of reduction in context to the d-dimensional case.

[2] This is not really a saying, but it is suitable here.

Chapter 4

When tired of counting, we can always enumerate

In this chapter, we introduce an algorithm that computes the set of d-concepts of a d-context. This algorithm takes as input a set of d-concepts from a d-context and a new element to be added to this d-context, and outputs the set of updated d-concepts. This incremental setup allows to update the set of concepts when the d-context changes, instead of computing again all the d-concepts from scratch. This solution allows to reduce the impact of the combinatorial explosion induced by the multidimensionality (sometimes called curse of dimensionality).

4.1 Motivation

Given a d-context, a classical task in Polyadic Concept Analysis is to enumerate all its d-concepts. This task finds applications in a lot of fields involving data. This is the classical enumeration problem addressed in its 2-dimensional form by Formal Concept Analysis with algorithms such as NEXT CLOSURE and we tackle it in Section 4.4, with Algorithm 2.

d-CONCEPT ENUMERATION

Input: A d-context $\mathcal{C}$.

Output: $\mathcal{T}(\mathcal{C})$, the set of all the d-concepts of $\mathcal{C}$.

In order to solve the d-CONCEPT ENUMERATION problem, we iteratively solve another problem, the d-CONCEPT AUGMENTATION problem, using Algorithm 1 (Section 4.4.2).

d-CONCEPT AUGMENTATION

Input: The set of d-concepts of a d-context $\mathcal{C}_S$ and a $(d-1)$-context $\mathcal{C}'$.

Output: The set of all the d-concepts of the new d-context built from $\mathcal{C}_S$ and $\mathcal{C}'$.

This is the enumeration problem corresponding to the case when a d-context is augmented with a new entry and we want to revise its set of d-concepts instead of computing it from scratch.

Existing methods

In [41], Loïc Cerf, Jérémy Besson, Céline Robardet and Jean-François Boulicaut present DATA PEELER, an algorithm that solves the d-CONCEPT ENUMERATION problem. They allow one to extract closed d-itemsets (essentially d-concepts) that satisfy some constraints. DATA PEELER uses a binary exploration technique, by dividing the search space and building a tree. It is, to our knowledge, the first algorithm that allows for the enumeration of closed d-itemsets. It does not, however, admit an incremental version.

In [42], Makhalova and Nourine introduce an algorithm to compute d-dimensional concepts. They present a recursive algorithm that fixes $d-2$ dimensions of a d-context and uses known algorithms to compute the 2-concepts of the contexts resulting from fixing $d-2$ dimensions. The recursive ascent merges concepts to go back to the original dimension. Their approach is incremental in the sense that they solve subproblems of decreasing dimensions. They do not propose an explicit algorithm to solve the d-CONCEPT AUGMENTATION problem. They use pairwise comparison of $(k-1)$-concepts to compute the k-concepts, with k varying from 2 to d. We avoid this pitfall by using what we call projection, which is defined in the next section.

4.2 Specific notations and projections

From now on, the first component (resp. dimension) of a d-concept (resp. d-context) will arbitrarily be called the *height* of the concept/context while all the other components/dimensions will be called the *width*. [1] For example, the 3-concept $(123, a, \alpha\beta)$ from Figure 4.1 has height 123 and width $(a, \alpha\beta)$. A d-concept $(X_1, \dots, X_d)$ has height X_1 and width $(X_2, \dots, X_d)$.

[1] While the height is usually a measure and as such, a number, we consider that the height of a concept/context is a set.

	a	b	c	a	b	c	a	b	c
α	×	×		×			×		
β	×			×			×	×	
γ	×			×		×		×	×
		1			2			3	

Figure 4.1: A 3-context $\mathcal{C} = (Numbers, Latin, Greek, \mathcal{R})$ where $Numbers = \{1, 2, 3\}$, $Latin = \{a, b, c\}$ and $Greek = \{\alpha, \beta, \gamma\}$.

Let $\mathcal{C}_S = (\mathcal{S}_1, \ldots, \mathcal{S}_d, \mathcal{R}_S)$ be a d-context and $\mathcal{C}' = (\mathcal{S}_2, \ldots, \mathcal{S}_d, \mathcal{R}')$ be a $(d-1)$-context on the same last $d-1$ dimensions as $\mathcal{C}_S$. The d-context $\mathcal{C}_S \oplus \mathcal{C}' = (\mathcal{S}_1 \cup \{e\}, \mathcal{S}_2, \ldots, \mathcal{S}_d, \mathcal{R}_S \cup \{(e, x_2, \ldots, x_d) \mid (x_2, \ldots, x_d) \in \mathcal{R}'\})$ is constructed by "gluing" $\mathcal{C}'$ on the bottom of $\mathcal{C}_S$, as illustrated in Figure 4.2. The element e, that is not in $\mathcal{S}_1$, serves as the identifier of $\mathcal{C}'$ on the first dimension (the one that $\mathcal{C}'$ lacks). Figure 4.3 depicts another way to visualise the gluing of $\mathcal{C}'$ on $\mathcal{C}_S$.

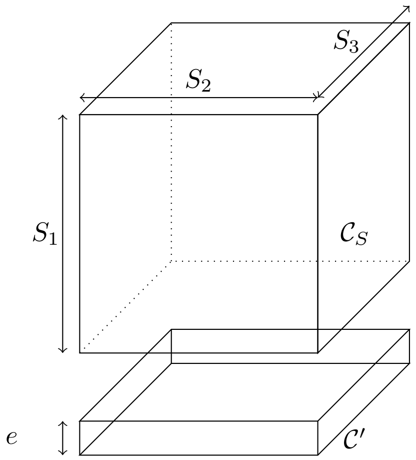


Figure 4.2: The 2-context $C' = (\mathcal{S}_2, \mathcal{S}_3, \mathcal{R}')$ corresponding to a new entry is glued on the bottom of the 3-context $\mathcal{C}_S = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{R}_S)$ by the $\oplus$ operation. Since this new entry does not have a first dimension corresponding to $\mathcal{S}_1$, we use e as its identifier.

Let H be a subset of the first dimension. The set $\mathcal{H}_H(\mathcal{C}_S)$ is the set of all the d-concepts of $\mathcal{C}_S$ of height H, i.e. all the d-concepts of the form $(H, X_2, \ldots, X_d)$ in $\mathcal{T}(\mathcal{C}_S)$.

An important notion to be manipulated in the algorithm is the *projection* of a d-concept onto a $(d-1)$-context.

Definition 42. *Let $\mathcal{C}_S$ be a d-context and let $\mathcal{C}'$ be a $(d-1)$-context whose dimensions are the width of $\mathcal{C}_S$. Let $X = (X_1, \ldots, X_d)$ be a d-concept of $\mathcal{C}_S$. The* projection *of X onto $\mathcal{C}'$, denoted by $\mathcal{P}_{\mathcal{C}'}(X)$, is the set $\mathcal{R}' \cap \prod_{i \in \{2, \ldots, d\}} X_i$.*

Let $\mathcal{X}$ be a set of d-concepts. Then the projection of $\mathcal{X}$ onto $\mathcal{C}'$ is the union of the

	$\mathcal{C}$											
	$\mathcal{C}_S$									$\mathcal{C}'$		
	a	b	c	a	b	c	a	b	c	a	b	c
α	×	×		×			×			×	×	
β	×			×			×	×			×	×
γ	×			×		×		×	×		×	×
		1			2			3			e	

Figure 4.3: By gluing the 2-context $\mathcal{C}'$ corresponding to e under our example $\mathcal{C}_S$, we obtain a new 3-context $\mathcal{C}$. Now that the entry e has been added, we have a d-context that may not have the same d-concepts.

projections of all the concepts in $\mathcal{X}$, i.e. $\mathcal{P}_{\mathcal{C}'}(\mathcal{X}) = \bigcup_{X \in \mathcal{X}} \mathcal{P}_{\mathcal{C}'}(X)$. A projection is a set of tuples that represent crosses.

Remark 43. *Let $\mathcal{X}$ be a set of d-concepts. Then $(\mathcal{S}_2, \ldots, \mathcal{S}_d, \mathcal{P}_{\mathcal{C}'}(\mathcal{X}))$ is a $(d-1)$-context.*

Figure 4.4 shows the projection of a d-concept of $\mathcal{C}_S$ (grey area in $\mathcal{C}_S$) onto a $(d-1)$-context $\mathcal{C}'$. The crosses that are in $\mathcal{R}'$ and in the width of the d-concept are shaded as well. Let us illustrate the projection by giving an example. We project the 3-concepts found in Figure 4.1 onto the 2-context induced by e in Figure 4.3:

$\mathcal{P}_{\mathcal{C}'}((12, a, \alpha\beta\gamma)) = \{(a, \alpha)\}$	$\mathcal{P}_{\mathcal{C}'}((1, ab, \alpha)) = \{(a, \alpha), (b, \alpha)\}$
$\mathcal{P}_{\mathcal{C}'}((2, ac, \gamma)) = \{(c, \gamma)\}$	$\mathcal{P}_{\mathcal{C}'}((123, a, \alpha\beta)) = \{(a, \alpha)\}$
$\mathcal{P}_{\mathcal{C}'}((3, ab, \beta)) = \{(b, \beta)\}$	$\mathcal{P}_{\mathcal{C}'}((3, b, \beta\gamma)) = \{(b, \beta), (b, \gamma)\}$
$\mathcal{P}_{\mathcal{C}'}((3, bc, \gamma)) = \{(b, \gamma), (c, \gamma)\}$	$\mathcal{P}_{\mathcal{C}'}((23, c, \gamma)) = \{(c, \gamma)\}$,
$\mathcal{P}_{\mathcal{C}'}((\emptyset, abc, \alpha\beta\gamma)) = \mathcal{R}'$	$\mathcal{P}_{\mathcal{C}'}((123, \emptyset, \alpha\beta\gamma)) = \emptyset$
$\mathcal{P}_{\mathcal{C}'}((123, abc, \emptyset)) = \emptyset$	

4.3 Important properties

Let $\mathcal{C} = \mathcal{C}_S \oplus C'$ be a d-context. Our goal is to solve the problems d-CONCEPT ENUMERATION and d-CONCEPT AUGMENTATION. This section focuses on properties needed to solve d-CONCEPT AUGMENTATION, that is to compute $\mathcal{T}(\mathcal{C})$ from $\mathcal{T}(\mathcal{C}_S)$ and $\mathcal{C}'$.

Proposition 44. *Let $X = (X_1, \ldots, X_d)$ be a d-concept of $\mathcal{C}$. Then, there exists a d-concept $X_S = (X_1 \setminus \{e\}, Y_2, \ldots, Y_d)$ in $\mathcal{C}_S$ with $X_i \subseteq Y_i$, $\forall i \in \{2, \ldots, d\}$.*

Proof. If $X = (X_1, \ldots, X_d)$ is a d-concept of $\mathcal{C}$, then it is a d-box full of crosses in $\mathcal{C}$. As such, removing e from X's height does not change the fact that $X_1 \setminus \{e\} \times X_2 \times \cdots \times X_d$ is a d-box full of crosses in $\mathcal{C}$. Consequently, $X_1 \setminus \{e\} \times X_2 \times \cdots \times X_d$ is also a d-box full of crosses in $\mathcal{C}_S$.

Moreover, in $\mathcal{C}$, $(X_1 \setminus \{e\}, X_2, \ldots, X_d)$ is maximal (inclusion-wise) for the height, as $(X_1, X_2, \ldots, X_d)$ would not be a d-concept of $\mathcal{C}$ otherwise. As $(X_1 \setminus$

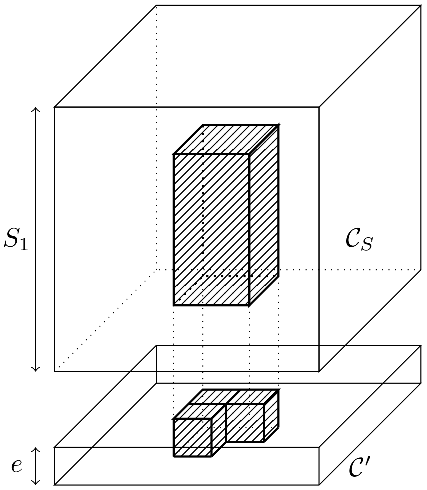


Figure 4.4: The $(d-1)$-context $(S_2, \ldots, S_d, \mathcal{P}_{\mathcal{C}'}(X))$ is constructed from $\mathcal{C}'$ by keeping only the crosses "under" X.

$\{e\}, X_2, \ldots, X_d)$ is not necessarily maximal on the other dimensions, it can be extended into a d-concept $X_S = (X_1 \setminus \{e\}, Y_2, \ldots, Y_d)$ of $\mathcal{C}_S$ with bigger width. □

Proposition 44 states that, for each d-concept C of $\mathcal{C}$, there is at least one d-concept in $\mathcal{C}_S$ that has a smaller (inclusion-wise) height and a bigger (inclusion-wise) width than C. Figure 4.5 shows a graphical example of Proposition 44.

In $\mathcal{C}$ (Figure 4.3), let us consider the 3-concept $(123e, a, \alpha)$. Proposition 44 states that there is a concept in $\mathcal{C}_S$ (Figure 4.1) that has height 123, and a larger width. In our example, $(123, a, \alpha\beta)$ is such a concept.

Corollary 45. *Let $X = (X_1, \ldots, X_d)$ be a d-concept of $\mathcal{C}$. If $e \notin X_1$, then X is a concept in $\mathcal{C}_S$.*

Clearly, when e is not contained in the height of a d-concept of $\mathcal{C}$, then this d-concept is untouched by the changes in the d-context, and as such was already a maximal box of crosses in the d-context $\mathcal{C}_S$.

In Figure 4.3, $(12, a, \alpha\beta\gamma)$ is a concept of both $\mathcal{C}_S$ and $\mathcal{C}$ that is unchanged by the addition of the new layer.

Using projection, we want to link the d-concepts of $\mathcal{C}_S$ to $\mathcal{C}'$. To this end, we introduce the following property, where the d-concepts of $\mathcal{C}_S$ are grouped by height.

Proposition 46. *Let $X = (X_1, \ldots, X_d)$ be a d-concept of $\mathcal{C}$, with e in its height. Then $(X_2, \ldots, X_d)$ is a $(d-1)$-concept of the $(d-1)$-context resulting from the projection $\mathcal{P}_{\mathcal{C}'}(\mathcal{H}_{X_1 \setminus \{e\}}(\mathcal{C}_S))$ of all the concepts of $\mathcal{C}_S$ of height $X_1 \setminus \{e\}$ onto $\mathcal{C}'$.*

Proof. Since $X = (X_1, \ldots, X_d)$ is a d-concept of $\mathcal{C}$, we know that $(X_2, \ldots, X_d)$ is a $(d-1)$-box of crosses in $\mathcal{C}'$. Moreover, from Proposition 44 we know that there is a d-concept Y of $\mathcal{C}_S$ with its width larger than $(X_2, \ldots, X_d)$.

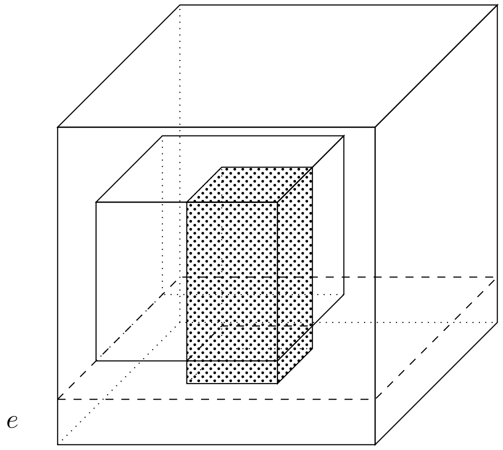


Figure 4.5: Let $X = (X_1, \ldots, X_d)$ be the dotted 3-concept of $\mathcal{C}$ that contains e in its height. Then there exist a 3-concept of $\mathcal{C}_S$ of height $X_1 \setminus \{e\}$. This concept has a wider width than X. Such a concept is drawn in this figure.

This ensures that $(X_2, \ldots, X_d)$ is a $(d-1)$-box full of crosses in the $(d-1)$-context $\mathcal{C}_P = (\mathcal{S}_2, \ldots, \mathcal{S}_d, \mathcal{P}_{\mathcal{C}'}(\mathcal{H}_{X_1 \setminus \{e\}}(\mathcal{C}_S)))$ resulting from the projection of all the d-concepts of $\mathcal{C}$ of height $X_1 \setminus \{e\}$.

For the sake of contradiction, let us now suppose that $(X_2, \ldots, X_d)$ is not maximal and thus not a $(d-1)$-concept of $\mathcal{C}_P$. Without loss of generality, suppose that it can be extended on its first component: there is a k in $\mathcal{S}_2$ such that $X_2 \cup \{k\} \times \cdots \times X_d$ is a $(d-1)$-box full of crosses in $\mathcal{C}_P$. This implies that $\mathcal{C}_S$ has a d-concept $(X_1 \setminus \{e\}, Z_2, \ldots, Z_d)$ where k is in Z_2 and $X_i \subseteq Z_i$ for all $i \in \{2, \ldots, d\}$. If this is the case, then X can be extended and is not a d-concept, which contradicts our premise. Thus, $(X_2, \ldots, X_d)$ is a $(d-1)$-concept of $\mathcal{C}_P$. □

Proposition 46 states that the width of the new d-concepts of height $H \cup \{e\}$ can be found among the $(d-1)$-concepts of the $(d-1)$-context resulting from the projections on $\mathcal{C}'$ of all the d-concepts of height H.

Let us go back once more to our example in Figure 4.3. In $\mathcal{C}$, $(3e, b, \beta\gamma)$ is a concept that contains e (which is our new element here) in its height. Proposition 46 states that the width of this 3-concept can be found among the 2-concepts of the 2-context resulting from the projection of all 3-concept of height $H = \{3\}$ onto $\mathcal{C}'$.

For example, let us compute $\mathcal{H}_{\{3\}}(\mathcal{C}_S) = \{(3, ab, \beta), (3, b, \beta\gamma), (3, bc, \gamma)\}$. Then the projection of these concepts onto $\mathcal{C}'$ is $\mathcal{P}_{\mathcal{C}'}(\mathcal{H}_{\{3\}}(\mathcal{C}_S)) = \{(b, \beta), (b, \gamma), (c, \gamma)\}$. The context induced by this projection is shown in Figure 4.6. The 2-concept $(b, \beta\gamma)$ is indeed a 2-concept of the 2-context induced by the projection of the 3-concepts of height 3, and $(3e, b, \beta\gamma)$ is a concept of $\mathcal{C}$.

Proposition 47. *Let $(X_2, \ldots, X_d)$ be a $(d-1)$-concept of the $(d-1)$-context*

$$(\mathcal{S}_2, \ldots, \mathcal{S}_d, \mathcal{P}_{\mathcal{C}'}(\mathcal{H}_H(\mathcal{C}_S)))$$

	a	b	c
α			
β		$\times$	
γ		$\times$	$\times$

e

Figure 4.6: The 2-context $(Greek, Latin, \mathcal{P}_{\mathcal{C}'}(\mathcal{H}_{\{3\}}(\mathcal{C}_S)))$, with the identifier e.

resulting from the projection of all the d-concepts of height H of $\mathcal{C}_S$ onto $\mathcal{C}'$. Then there is a d-concept $(Y, X_2, \ldots, X_d)$ of $\mathcal{C}$ with $H \cup \{e\} \subseteq Y$.

Proof. If $(X_2, \ldots, X_d)$ is a $(d-1)$-concept of $\mathcal{C}_H = (\mathcal{S}_2, \ldots, \mathcal{S}_d, \mathcal{P}_{\mathcal{C}'}(\mathcal{H}_H(\mathcal{C}_S)))$, then there are d-concepts of the form $(H, Y_2, \ldots, Y_d)$ in $\mathcal{C}_S$, with $X_i \subseteq Y_i$, for all $i \in \{2, \ldots, d\}$. If, because the height is not maximal, $(H \cup \{e\}, X_2, \ldots, X_d)$ is not a concept of $\mathcal{C}$, then there exist d-concepts of the form $(Z, Z_2, \ldots, Z_d)$ in $\mathcal{C}_S$ with $H \subset Z$ and $X_i \subseteq Z_i$ for all $i \in \{2, \ldots, d\}$. Thus, $X_2 \times \cdots \times X_d$ is a $(d-1)$-box full of crosses in the context that results from the projection $\mathcal{P}_{\mathcal{C}'}(\mathcal{H}_Z(\mathcal{C}_S))$.

If $(X_2, \ldots, X_d)$ is not a $(d-1)$-concept of $\mathcal{C}_Z = (\mathcal{S}_2, \ldots, \mathcal{S}_d, \mathcal{P}_{\mathcal{C}'}(\mathcal{H}_Z(\mathcal{C}_S)))$, then it means that it can be extended. Without loss of generality, let us say that it can be extended on its first component, i.e. there exists a k such that $X_2 \cup \{k\} \times \cdots \times X_d$ is a subset of $\mathcal{P}_{\mathcal{C}'}(\mathcal{H}_Z(\mathcal{C}_S))$. This implies that $Z \times \{k\} \times X_3 \times \cdots \times X_d$ is in $\mathcal{R}_S$ and thus $H \times \{k\} \times X_3 \times \cdots \times X_d$ is in $\mathcal{R}_S$. This contradicts the fact that $(X_2, \ldots, X_d)$ is a $(d-1)$-concept of C_H, as it could be extended on k. Consequently, $(X_2, \ldots, X_d)$ must be a $(d-1)$-concept of $\mathcal{C}_Z$.

To conclude, if $(X_2, \ldots, X_d)$ is a $(d-1)$-concept of $\mathcal{C}_H$, then either $(H \cup \{e\}, X_2, \ldots, X_d)$ is a concept of $\mathcal{C}$ or $(X_2, \ldots, X_d)$ is a $(d-1)$-concept of the $(d-1)$-context $(\mathcal{S}_2, \ldots, S_d, \mathcal{P}_{\mathcal{C}'}(\mathcal{H}_Z(\mathcal{C}_S)))$ for some superset Z of H. Eventually, because $\mathcal{S}_1$ is finite, a maximal height M such that $(M \cup \{e\}, X_2, \ldots X_d)$ is a concept of $\mathcal{C}$ is reached. □

Proposition 47 implies that every $(d-1)$-concept that appears in the $(d-1)$-context resulting from the projection of a given height on $\mathcal{C}'$ will eventually be the width of some d-concept.

Let us consider our running example, shown in Figure 4.3, and the 2-context of Figure 4.6 resulting from the projection of all the 3-concepts of height $\{3\}$ of the running example. The 2-concepts in Figure 4.6 are $(b, \beta\gamma)$ and (bc, γ). Both those 2-concepts give rise to new 3-concepts in the augmented context: $(3e, b, \beta\gamma)$ and $(3e, bc, \gamma)$.

From Propositions 44, 46 and 47, we can infer how to compute the new d-concepts that contain $\{e\}$ in their height by projecting the "old" d-concepts.

Remark 48. *Some d-concepts from $\mathcal{T}(\mathcal{C}_S)$ will not be in $\mathcal{T}(\mathcal{C})$: if $X = (X_1, \ldots, X_d)$ is a d-concept of $\mathcal{C}_S$ and $\mathcal{P}_{\mathcal{C}'}(X) = X_2 \times \cdots \times X_d$, then Proposition 46 implies that $(X_1 \cup \{e\}, X_2 \ldots, X_d)$ will be a d-concept of $\mathcal{C}$ and replace X.*

4.4 Presentation of the algorithm

In this section, we present an algorithm to address the d-CONCEPT AUGMENTATION problem (Section 4.4.1), and then an algorithm to address the d-CONCEPT ENUMERATION problem (Section 4.4.2). The second algorithm relies on the first to compute the set of all the d-concepts of a d-context progressively.

4.4.1 d-CONCEPT AUGMENTATION

Propositions 44, 46 and 47 give us the tools necessary to compute $\mathcal{T}(\mathcal{C})$ from $\mathcal{T}(\mathcal{C}_S)$ and $\mathcal{C}'$. We use them in Algorithm 1.

First, for every height H that corresponds to at least one d-concept of $\mathcal{C}_S$, we compute $\mathcal{H}_H(\mathcal{C}_S)$, and the $(d-1)$-context $(\mathcal{S}_2, \ldots, \mathcal{S}_d, \mathcal{P}_{\mathcal{C}'}(\mathcal{H}_H(\mathcal{C}_S)))$ resulting from the projection of $\mathcal{H}_H(\mathcal{C}_S)$ onto $\mathcal{C}'$.

For each such $(d-1)$-context, we compute its set of $(d-1)$-concepts. These $(d-1)$-concepts are then transformed into d-concept candidates by adding $\{e\}$ to the height that generated the $(d-1)$-context from which they originate. Finally, these candidates are checked for maximality on the height and added to the new set of d-concepts if needed. The removal of d-concepts that are no longer maximal occurs during the computation of their projection on $\mathcal{C}'$ (cf Remark 48).

```
Algorithm 1: $d$-CONCEPT AUGMENTATION($\mathcal{T}(\mathcal{C}_S), \mathcal{C}'$)
Input: $\mathcal{T}(\mathcal{C}_S)$ the set of $d$-concepts of $d$-context $\mathcal{C}_S$ and $\mathcal{C}'$ a $(d-1)$-context.
Output: $\mathcal{T}(\mathcal{C})$ the set of $d$-concept of $\mathcal{C} = \mathcal{C}_S \oplus \mathcal{C}'$.
$R \leftarrow \mathcal{T}(\mathcal{C}_S)$
foreach different heights $H$ do
    $P \leftarrow \emptyset$
    foreach $X = (H, X_2, \ldots, X_d)$ in $\mathcal{H}_H(\mathcal{C}_S)$ do
        $P \leftarrow P \cup \mathcal{P}_{\mathcal{C}'}(X)$
        if $\mathcal{P}_{\mathcal{C}'}(X) = \prod_2^d X_i$ then
            $R \leftarrow R \setminus \{X\}$
    Compute $\mathcal{T}(\mathcal{S}_2, \ldots, \mathcal{S}_d, P)$
    foreach $(Y_2, \ldots, Y_d) \in \mathcal{T}(\mathcal{S}_2, \ldots, \mathcal{S}_d, P)$ do
        if $(H \cup \{e\}, Y_2, \ldots, Y_d)$ is maximal on its height then
            $R \leftarrow R \cup (H \cup \{e\}, Y_2, \ldots, Y_d)$
return $R$
```

Proposition 49. *Algorithm 1 ends and returns each d-concept of $\mathcal{C}$ exactly once.*

Proof. The dimensions of the contexts are finite. The set of d-concepts of a d-context is finite too and so is the number of possible heights. The algorithm goes through each height and projects each d-concept once. Thus, Algorithm 1 ends.

Proposition 46 ensures that every d-concept $(H \cup \{e\}, X_2, \dots, X_d)$ is found when the loop processes the height H so every d-concept is found at least once. As a d-concept $(H \cup \{e\}, X_2, \dots, X_d)$ can only be generated for the height H, every d-concept is added to the output only once. □

4.4.2 d-CONCEPT ENUMERATION

In this section, our goal is to solve the d-CONCEPT ENUMERATION problem, using the approach of Algorithm 1. We recall the problem.

d-CONCEPT ENUMERATION

Input: A d-context $\mathcal{C}$.

Output: All the d-concepts of $\mathcal{C}$.

Computing the d-concepts of a d-context from scratch can be done with Algorithm 1 by starting with a single d-concept $(\emptyset, \mathcal{S}_2, \dots, S_d)$ and adding the layers corresponding to $\mathcal{S}_1$ one by one, using in Algorithm 2. As per Remark 52, adding a layer cannot reduce the number of concepts. Computing $\mathcal{T}(\mathcal{C})$ from scratch is in $\mathcal{O}\left(|\mathcal{S}_1| \times Q\right)$ where Q is the complexity of Algorithm 1.

To ease the notations, we introduce the following definition.

Definition 50. *Let $\mathcal{C} = (\mathcal{S}_1, \dots, \mathcal{S}_d, \mathcal{R})$ be a d-context. Let s be an element of $\mathcal{S}_1$. The $(d-1)$-context $\mathcal{C}^s = (S_2, \dots, S_d, \mathcal{R}^s)$ where $R^s = \{(x_2, \dots, x_d) \mid (s, x_2, \dots, x_d) \in \mathcal{R}\}$ is obtained by keeping only the layer corresponding to s.*

Algorithm 2: d-CONCEPT ENUMERATION($\mathcal{C}$)

```
Input: $\mathcal{C} = (\mathcal{S}_1, \dots, \mathcal{S}_d, \mathcal{R})$ a $d$-context.
Output: $\mathcal{T}(\mathcal{C})$ the set of $d$-concepts of $\mathcal{C}$.
if $d = 1$ then
    $C \leftarrow \{\mathcal{R}\}$
else
    $C \leftarrow (\emptyset, S_2, \dots, S_d)$
    foreach $s \in \mathcal{S}_1$ do
        $C \leftarrow d$-CONCEPT AUGMENTATION$(C, \mathcal{C}^s)$
return $C$
```

Algorithm 2 can be used to compute the $(d-1)$-concepts of the projections in Algorithm 1, which makes it recursive. The recursion ends when reaching 1-contexts $(\mathcal{S}_1, R)$ which have single 1-concepts that can be computed in $O(|\mathcal{S}_1|)$.

4.4.3 Complexity

Since the output of our problem may be exponential in the size of its input, no polynomial algorithm[2] can exist. In order to study the complexity of Algorithm 1, we will use the notion of output-polynomiality.

Definition 51. *An algorithm is said to be* output-polynomial *if, for all input I, its time complexity is in $\mathcal{O}((|I| + |O|)^k)$ where $|O|$ is the size of the output and k a constant.*

It is clear that we need to study the running time of our algorithm not only in terms of size of the input, but also in relation to the size of the output. We recall the input and output of d-CONCEPT AUGMENTATION.

d-CONCEPT AUGMENTATION

Input: The set of d-concepts of a d-context $\mathcal{C}_S$ and a $(d-1)$-context $\mathcal{C}'$.

Output: $\mathcal{T}(\mathcal{C}_S \oplus \mathcal{C}')$, the set of all the d-concepts of the d-context $\mathcal{C}_S \oplus \mathcal{C}'$.

At the time of writing, the only known bound for the number of d-concepts of a d-context $(\mathcal{S}_1, \dots, \mathcal{S}_d, \mathcal{R})$ is $\prod_{i\in\{1,\dots,d\}\setminus\{k\}} 2^{|\mathcal{S}_i|}$ with $k = argmax_{k\in\{1,\dots,d\}}|\mathcal{S}_k|$. This comes from the fact that a d-concept is uniquely described by $d-1$ of its components. By supposing that all the subsets of the $d-1$ smallest dimensions can appear in d-concepts (and that is a bold assumption), we have our bound. [3]

The size of a d-context $(\mathcal{S}_1, \dots, \mathcal{S}_d, \mathcal{R})$ is $\prod_{i\in\{1,\dots,d\}} |\mathcal{S}_i|$. Our input is a set of d-concepts and a $(d-1)$-context. We denote by I the input. It's size is bounded by $\prod_{i\in\{1,\dots,d\}\setminus\{k\}} 2^{|\mathcal{S}_i|} + \prod_{i\in\{2,\dots,d\}} |\mathcal{S}_i|$. Since our output O is a set of d-concepts, its size is bounded by $\prod_{i\in\{1,\dots,d\}\setminus\{k\}} 2^{|\mathcal{S}_i|}$.

Remark 52. *It is to be noted that $|\mathcal{T}(\mathcal{C}_S)| \leq |\mathcal{T}(\mathcal{C})|$.*

Proof. If a d-concept exists in $\mathcal{C}_S$, it is either broken into multiple "thinner" concepts, kept intact or extended with the new element in $\mathcal{C}$. □

The number of heights is bounded by $2^{|\mathcal{S}_1|}$ and $|\mathcal{T}(\mathcal{C}_S)|$. As per Proposition 47, a d-concept appearing in a $(d-1)$-context resulting from the projection of the d-concepts of a given height is always the width of some d-concept in $\mathcal{C}$. Such a d-concept cannot appear in more than $min\left(2^{|\mathcal{S}_1|}, \mathcal{T}(\mathcal{C}_S)\right)$ $(d-1)$-contexts. From this follows that the algorithm will not handle more than $min\left(2^{|\mathcal{S}_1|}, \mathcal{T}(\mathcal{C}_S)\right) \times |\mathcal{T}(\mathcal{C}) \setminus \mathcal{T}(\mathcal{C}_S)|$ $(d-1)$-concepts.

Computing $\mathcal{H}_H(\mathcal{C}_S)$ for every possible height – i.e. grouping the d-concepts by height – can be done using a sorting algorithm in $\mathcal{O}\left(|\mathcal{T}(\mathcal{C}_S)| \times \log|\mathcal{T}(\mathcal{C}_S)|\right)$. The projection of a d-concept $(X_1, \dots, X_d)$ onto $\mathcal{C}'$ is computed in $\prod_{i\in\{2,\dots,d\}} |X_i|$.

As mentioned previously, the $(d-1)$-concepts manipulated by the algorithm are the width of some d-concepts, hence they are maximal. As such, only the height has to be tested for maximality in order to decide whether a d-box is a d-concept. The complexity of doing so depends on the information available. If we know $\mathcal{C}_S$ (and not

[2] In the traditional sense: polynomial in the size of the input.

[3] That's a debatable statement seeing the result in Chapter 2.

only its set of d-concepts), deciding whether $(H, X_2, \ldots, X_d)$ is a d-concept can be done in $\mathcal{O}(|\mathcal{S}_1 \setminus H| \times |X_2| \times \cdots \times X_d)$ by checking whether a layer that is not in H contains the width. If we do not have $\mathcal{C}_S$, we can use the fact that $(H, X_2, \ldots, X_d)$ is a d-concept if and only if $H \setminus \{e\}$ is the greatest height (inclusion-wise) for which $(X_2, \ldots, X_d)$ is a $(d-1)$-concept in $\mathcal{P}_{\mathcal{C}'}(\mathcal{H}_{H\setminus\{e\}}(\mathcal{C}_S))$. By sorting the concepts by their heights into any linear order such that $H_i \leq H_j \Leftrightarrow H_j \subseteq H_i$, a new d-concept will be created as soon as its width is found for the first time.

We can check whether a concept $(X_2, \ldots, X_d)$ is already the width of a concept of greater height by storing old concepts in a hashtable structure. Hashtable structures can ensure an average $O(1)$ cost for the look-up of a concept. However, as the universe of keys is $2^{|X_2|+\cdots+|X_d|}$, and the number of stored concept big, in order to keep the load factor under a certain threshold, one would need to allocate $O(2^{|X_2|+\cdots+|X_d|})$ in memory space.

Proposition 53. *The algorithm that computes $\mathcal{T}(\mathcal{C})$ from $\mathcal{T}(\mathcal{C}_S)$ and $\mathcal{C}'$ is in*

$$\mathcal{O}\left(\underbrace{|\mathcal{T}(\mathcal{C}_S)| \times \left(\log|\mathcal{T}(\mathcal{C}_S)| + \prod_{i\in\{2,\ldots,d\}} |\mathcal{S}_i|\right)}_{\text{Sort and project the } d\text{-concepts of } \mathcal{C}_S} + \right.$$

$$\left. \underbrace{L \times (K + |\mathcal{T}(\mathcal{C}) \setminus \mathcal{T}(\mathcal{C}_S)| \times M)}_{\text{Processing for each height}} \right)$$

with L the number of possible height (bounded by $2^{|\mathcal{S}_1|}$ and $|\mathcal{T}(\mathcal{C}_S)|$), K the complexity of computing the $(d-1)$-concepts of a $(d-1)$-context and M the complexity of checking whether a d-box is a d-concept.

Proposition 54. *Algorithm 1 enumerates the elements of $\mathcal{T}(\mathcal{C}^e)$ in output-polynomial time.*

Proof. The number of possible heights is less than $min(2^{|S_1|}, \mathcal{T}(\mathcal{C}_S))$. Checking whether a d-box is a d-concept can be done in time polynomial in the size of the d-context. From Proposition 47, we know that the number of $(d-1)$-concepts in a $(d-1)$-context resulting from projections on $\mathcal{C}'$ is bounded by the size of the output.

As such, computing the d-concepts of $\mathcal{T}(\mathcal{C})$ from those of $\mathcal{T}(\mathcal{C}_S)$ and $\mathcal{C}'$ can be done in output-polynomial time if and only if computing the $(d-1)$-concepts of the projections can also be done in output-polynomial time. Computing all the 1-concepts of a 1-context can be done in polynomial time so computing the 2-concepts of a 2-context can be done in output-polynomial time. Recursively, we can see that computing the d-concepts of a d-context can also be done in output-polynomial time. □

Conclusion

In this chapter we introduced a new incremental algorithm to retrieve the d-concepts of a d-context. It does not allow for the computation order structure of the subjacent d-lattice. As new d-concepts are constructed by adding a single element to the height of another concept, it is easy to retrieve a spanning tree of the covering relation of the partial order induced by the inclusion on the first dimension. The same cannot be said for all the other dimensions. While applications requiring only the spanning tree of $\leq_1$ are likely to exist, it would be interesting to be able to compute the whole lattice, either by modifying Algorithm 1 or through a new approach.

Though not the first incremental algorithm for this problem, Algorithm 1 is the first not to require the initial context ($\mathcal{C}_S$). Even so, it would be interesting to experimentally compare the running time of these algorithms, both on real life datasets and generated datasets. We hope that this algorithm may round off the algorithmic toolbox available in d dimensions.

Chapter 5

Let's shrink the structures

As we have seen in the previous chapters, one of the limitations of polyadic concept analysis lies in the combinatorial explosion that comes with d-contexts and d-lattices. This limitation forces us to consider approaches that avoid the combinatorial explosion. For example, in Chapter 4, we introduced an incremental algorithm in order to be able to speed-up the computation of d-concepts following an addition to a d-dimensional dataset.

Another way around the combinatorial limitation due to d-dimensional data considers not the algorithms that extract d-concepts, but the dataset itself. This approach is called *reduction*. It is well known in the 2-dimensional case and has been extended to triadic datasets by Rudolph, Săcărea and Troancă [43]. Here, we extend the notion of reduction to the d-dimensional case. Reduction in a dataset allows one to get rid of some "*useless*" entries. Here, the word useless is between quotes and in italics because the value of an entry of the dataset is different depending on the goal one wants to achieve. When only the underlying lattice is of importance, reduction changes nothing, and reducible entries are, indeed, useless. When one wants to really extract some knowledge of a dataset, the loss of some entries, even if those are elements that do not change the lattice, is tragic. That is why we consider the next strategy.

The third way of overcoming limitations due to the dimension and the size of the dataset that we consider in this dissertation is the restriction of the set of computed concepts. Instead of considering all the d-concepts of a d-context, some applications that do not tolerate the loss of information implied by reduction are compatible with another approach: the computation of introducer concepts. Introducer concepts are well studied in 2-dimensions, where they received considerable attention under the name of AOC-posets or Galois sub-hierarchies. They are widely used in applications, including in this dissertation in Chapter 6. In this chapter, we define the set of introducer concepts in d-dimensions and give some insight on the structure of this set and its combinatorics.

5.1 Reduction in d-contexts

5.1.1 Reduction in 2-contexts

A context can be constructed from the set of its concepts in the following way: the set of objects is the extent of the largest concept, the set of attributes is the intent of the smallest concept, and the relation between them is the union of all pairs (a, b) such that (A, B) is a concept of the context and a is an element of A, and b an element of B. This relation between a lattice and its representation as a context is already hinted at in [2, Chapter v].

On the other hand, two concept lattices from different contexts can be isomorphic to each other. Reduction is a way of reaching a canonical context, the standard context, for any finite lattice. The first step of reduction in 2-dimension is the fusion of identical rows or columns.

Definition 55 (Reformulated from [3, Definition 23])**.** *A context $(\mathcal{S}_1, \mathcal{S}_2, \mathcal{R})$ is called **clarified** if for any objects o_1 and o_2 in S_1, $o'_1 = o'_2$ implies that $o_1 = o_2$. Correspondingly, for any attributes a_1 and a_2 in $\mathcal{S}_2$, $a'_1 = a'_2$ implies $a_1 = a_2$.*

In [3, Definition 23], the authors use the following example of a context that represent the service offers of an office supplies business an the associated clarified context (Figure 5.1).

Another possible action is the removal of attributes (resp. objects) that can be written as combination of other attributes (resp. objects). This is the proper definition of reducible attribute (resp. object). If a is an attribute and A is a set of attributes that does not contain a but have the same extent, then a is **reducible**. By definition, full rows and full columns are always reducible.

Definition 56 (Reformulated from [3, Definition 24])**.** *A context $(\mathcal{S}_1, \mathcal{S}_2, \mathcal{R})$ is called **row reduced** if every object-concept is $\vee$-irreducible and **column reduced** if every attribute-concept is $\wedge$-irreducible. A context that is both row reduced and column reduced is **reduced**.*

This yields that for every finite lattice L, there is a unique (up to an isomorphism) reduced context such that L is the concept lattice of this context. This context is called the standard context of L. This standard context can be obtained from any finite context by first clarifying the context and then deleting all the objects that can be represented as intersection of other objects and attributes that can be represented as intersection of other attributes.

5.1.2 Reduction in d-dimensions

To define reduction in multidimensional context, we need to recall some definitions. Let $\pi = (i, \{1, \ldots, d\} \setminus i)$ be a binary partition of $\{1, \ldots, d\}$, the context $\mathcal{C} =$

	Furniture	Computers	Copy-machine	Typewriters	Specialized machines
Consulting	×	×	×	×	×
Planning	×	×			
Installation	×	×	×	×	×
Instruction		×	×	×	×
Training		×			
Spare parts	×	×	×	×	×
Repairs	×	×	×	×	×
Service contracts		×	×	×	

	Furniture	Computers	Copy-machine and Typewriters	Specialized machines
Consulting, Installation, Spare parts, repairs	×	×	×	×
Planning	×	×		
Instruction		×	×	×
Training		×		
Service contracts		×	×	

Figure 5.1: Context and clarified context

$(\mathcal{S}_1, \dots, \mathcal{S}_d, \mathcal{R})$ gives rise to the context $\mathcal{C}^\pi = (\mathcal{S}_i, \prod_{j \in \{1,\dots,d\} \setminus \{i\}} \mathcal{S}_j, \mathcal{R}^\pi)$ where $(a, b) \in \mathcal{R}^\pi$ if and only if $a \times b_1 \times b_{d-1} \in \mathcal{R}$ and $b = b_1 \times b_{d-1}$. We refer the reader to figures 4.1 and 1.10 for a graphical representation of this transformation. Such a binary partition gives rise to the derivation operators $X \mapsto X^{(\pi)}$ on the 2-context $\mathcal{C}^\pi$.

We first defined clarified contexts. In the same way that in the 2-dimensional case, our definition is equivalent to the fusion of identical $(d-1)$-dimensional layers.

Definition 57. *A d-context $(\mathcal{S}_1, \dots, \mathcal{S}_d, \mathcal{R})$ is called* ***clarified*** *is for all i in $\{1, \dots, d\}$, for any x_1 and x_2 in $\mathcal{S}_i$, $x_1^{(\pi)} = x_2^{(\pi)}$ implies $x_1 = x_2$, with $\pi = (i, \{1, \dots, d\} \setminus i)$.*

As with the 2-dimensional case, we provide an example in Figure 5.2.

Definition 58. *A clarified d-context $\mathcal{C} = (\mathcal{S}_1, \dots, \mathcal{S}_d, \mathcal{R})$ is called i-reduced if every object-concept from $\mathcal{C}^{(\pi)}$, with $\pi = (i, \{1, \dots, d\} \setminus \{i\})$ is $\vee$-irreducible. A d-context is* ***reduced*** *if it is i-reduced for all i in $\{1, \dots, d\}$.*

Proposition 59. *Let $\mathcal{C} = (\mathcal{S}_1, \dots, \mathcal{S}_d, \mathcal{R})$ be a d context. Let π be a binary partition $(i, \{1, \dots, d\} \setminus \{i\})$. Let y_i be an element of dimension i and Y_i be a set of elements of dimension i such that y_i is not in Y_i and $y_i^{(\pi)} = Y_i^{(\pi)}$. Then,*

$$\mathcal{T}(\mathcal{C}) \cong \mathcal{T}\left(\left(\mathcal{S}_1, \dots, \mathcal{S}_i \setminus \{x_i\}, \dots, \mathcal{S}_d, \mathcal{R} \cap \left(\mathcal{S}_i \setminus \{x_i\} \times \prod_{j \in \{1,\dots,d\} \setminus \{i\}} \mathcal{S}_j\right)\right)\right).$$

Proof. Without loss of generality, let us assume that x_i is an element of $\mathcal{S}_1$. We have to ensure that if $(X_1, \dots, X_d)$ is a concept of $\mathcal{C}$, then $(X_1 \setminus \{x_i\}, \dots, X_d)$ is a

	a	b	c	a	b	c	a	b	c
α	×	×		×	×				
β			×	×	×	×			×
γ	×	×					×	×	×
		1			2			3	

	$(\alpha, 1)$	$(\alpha, 2)$	$(\alpha, 3)$	$(\beta, 1)$	$(\beta, 2)$	$(\beta, 3)$	$(\gamma, 1)$	$(\gamma, 2)$	$(\gamma, 3)$
a	×	×			×		×		×
b	×	×			×		×		×
c				×	×	×			×

	a and b	c	a and b	c	a and b	c
α	×		×			
β		×	×	×		×
γ	×				×	×
	1		2		3	

Figure 5.2: A 3-context $\mathcal{C} = (Numbers, Greek, Latin, \mathcal{R})$ (above) and the 2-context $\mathcal{C}^{\pi}$ with $\pi = (Latin, \{Greek, Numbers\})$. We can see that $a^{(\pi)} = b^{(\pi)} = \{(\alpha, 1), (\alpha, 2), (\beta, 2), (\gamma, 1), (\gamma, 3)\}$, which means that a and b can be aggregated into a new attribute "a and b". Below, the corresponding clarified 3-context.

concept in the reduced context. In $\mathcal{C}$, $(X_1 \setminus \{x_i\}, \ldots, X_d)$ is a d-box full of crosses. We have to show that removing x_i from X_1 does not allow it to be extended on any other dimension. As $(\mathcal{S}_1, \ldots, \mathcal{S}_d, \mathcal{R})$ is a d-concept, its components X_j, $j \neq i$ form a $(d-1)$-concept in the intersection of all the layers induced by the elements of dimension i. As x_i is reducible, $\mathcal{C}^{x_i}$ is the intersection of at least two layers $\mathcal{C}^a$ and $\mathcal{C}^b$. Obviously, a and b are elements of X_1. If $(X_2, \ldots, X_d)$ is not a $(d-1)$-concept in the intersection pf the layers $\mathcal{C}^x$, $x \in X_1 \setminus \{x_i\}$, then it can be extended using crosses that both $\mathcal{C}^a$ and $\mathcal{C}^b$ share. This means that $\mathcal{C}^{x_i}$ should have them too, preventing $(\mathcal{S}_1, \ldots, \mathcal{S}_d, \mathcal{R})$ from being a concept in the first place. This ensures that $(X_1 \setminus \{x_i\}, \ldots, X_d)$ is indeed a concept. □

This proposition states that removing a reducible element from a d-context does not change the structure of the underlying d-lattice. If we keep track of the reduced and clarified elements during the process, it is possible not to lose information (by creating aggregate attributes or objects for example). It is still a deletion from the dataset. In the next section we speak about introducer concepts in a multidimensional setting.

5.2 Introducer concepts

Reduction induces a loss of information in a dataset since reducible elements are erased. Lots of applications cannot afford this loss of information and have to use other ways of reducing the complexity. A new structure, smaller than the concept lattice, has been introduced by Godin and Mili [44] in 1993. This structure consists in the restriction of the lattice to the set of introducer concepts. In the general case, the properties that make a concept lattice a lattice are lost when restricting the set of concepts to the introducers. The set of introducers can still be ordered by inclusion on either of their dimensions to obtain a poset. Since dyadic FCA deals with objects and attributes, such a poset is also called an Attribute Object Concept poset, or AOC-poset for short. In this section we introduce the introducers in a d-dimensional setting.

First of all, let us recall the definition of object-concepts and attributes-concepts.

Definition 60. *Let $C = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{R})$ be a context. Let o be an element of $\mathcal{S}_1$. Then $(\{o\}'', \{o\}')$ is an object-concept, called the introducer of object o.*

Definition 61. *Let $C = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{R})$ be a context. Let a be an element of $\mathcal{S}_2$. Then $(\{a\}', \{a\}'')$ is an object-concept, called the introducer of attribute a.*

Due to the unicity of the dyadic closure (one component of a concept lets only one choice for the other), each element of the dimensions have only one introducer. This allows to bound the size of an AOC-poset by the number of objects plus the number of attributes of a context, when a concept lattice can have up to an exponential size in the number of objects or attributes. As we will see in the following, this property is lost when we go multidimensional.

Definition 62. *Let i be a dimension called the height while all other dimensions are called the width. Let x be an elements of dimension i. The concepts with maximal width such that x is in the height are the introducer concepts of x. The set of introducer concepts of x is denoted I_x.*

	a	b	c	a	b	c	a	b	c	a	b	c
α	×	×		×			×			×	×	
β	×			×			×	×			×	×
γ	×			×		×		×	×		×	×
		1			2			3			4	

Figure 5.3: This 3-context shall serve as an example of our definitions.

Let us consider the 3-context from Figure 5.3 as an example. Let us compute the introducers of element a. We have $I_a = \{(12, \alpha\beta\gamma, a), (123, \alpha\beta, a), (1234, \alpha, a)\}$. For the element 3, we have $I_3 = \{(123, \alpha\beta, a), (3, \beta, ab), (34, \beta\gamma, b), (34, \gamma, bc)\}$.

We denote by $\mathcal{I}(\mathcal{S}_i)$ the union of the introducer concepts of all the elements of a dimension i and by $\mathcal{I}(\mathcal{C})$ the set of all the introducer concepts of a context $\mathcal{C}$.

Proposition 63. *$(\mathcal{I}(\mathcal{C}), \lesssim_1, \ldots, \lesssim_d)$ is a d-ordered set.*

Proof. Let $A = (A_1, \ldots, A_d)$ and $B = (B_1, \ldots, B_d)$ be in $\mathcal{I}(\mathcal{C})$. We recall that $A_i \subseteq B_i \Leftrightarrow A \leq_i B$ and that $A_i = B_i \Leftrightarrow A \sim_i B$. Without loss of generality, A is an introducer for an element of dimension i and B for an element of dimension j. If, for all k between 1 and d, $A \sim_k B$, then for all k, $A_k = B_k$ and $A = B$ (Uniqueness Condition).

If A and B are distinct, then there exists a dimension k such that $A \lesssim_k B$ or $B \lesssim_k A$. Without loss of generality, let us assume the former. Suppose that there is no ℓ between 1 and d such that $A \lesssim_\ell B$. That implies that all the components of A are included in the components of B, which is a contradiction with the maximality condition implied by A being a concept. Thus, there exists a h in $\{\{1, \ldots, d\}\} \setminus \{k\}$ such that $B \lesssim_h A$ (Antiordinal Dependency). □

From now on we can mirror the terminology of the 2-dimensional case, where we have complete lattices and attribute-objects-concepts partially ordered set and use complete d-lattices and introducers d-ordered sets.

The following proposition links introducer d-concepts with the $(d-1)$-concepts that arise on a layer of a d-context.

Proposition 64. *Let x be an element of dimension i. If $(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_d)$ is a $(d-1)$-concept of $\mathcal{C}^x$, then there exists some X_i such that $(X_1, \ldots, X_{i-1}, X_i, X_{i+1}, \ldots, X_d)$ is an introducer of x. If $(X_1, \ldots, \{x\} \cup X_i, X_d)$ is an introducer of x, then there exists a $(d-1)$-concept $(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_d)$ in $\mathcal{C}^x$.*

Proof. We suppose, without loss of generality, that x is in $\mathcal{S}_1$. The $(d-1)$-concepts of $\mathcal{C}^x$ are of the form $(X_2, \ldots, X_d)$. If $(x, X_2, \ldots, X_d)$ is a d-concept of $\mathcal{C}$, then it is maximal in width and has x in its height, so it is an introducer of x.

If $(x, X_2, \ldots, X_d)$ is not a d-concept of $\mathcal{C}$, it means that it can be augmented only on the first dimension (since $(X_2, \ldots, X_d)$ is maximal in $\mathcal{C}^x$). Thus, there exists a d-concept $(\{x\} \cup X_1, X_2, \ldots, X_d)$ that is maximal in width and has x in its height and is, as such, an introducer for x.

Suppose that there is a $X = (X_1, \ldots, X_d)$ that is an introducer of x but that is not obtained from a $(d-1)$-concept of $\mathcal{C}^x$ by extending X_1. It means that $(X_2, \ldots, X_d)$ is not maximal in $\mathcal{C}^x$, else it would be a $(d-1)$-concept). Then there exists a d-concept $Y = (Y_1, Y_2, \ldots, Y_d)$ with $x \in Y_1 \subseteq X_1$ and $X_i \subseteq Y_i$ for all i between 1 and d. This is a contradiction with the fact that X is an introducer of x. □

Proposition 64 states that every $(d-1)$-concept of a layer $\mathcal{C}^x$ maps to an introducer of x in $\mathcal{C}$ and that every introducer of x is the image of a $(d-1)$-concept of $\mathcal{C}^x$. This proposition gives a naive algorithm to compute the set of introducer concepts of a d-context. It is sufficient to compute the $(d-1)$-concepts of the $(d-1)$-contexts obtained by fixing an element of a dimension.

Algorithm 3 computes the introducers for each element of a dimension i. For a given element $x \in \mathcal{S}_i$, we compute $\mathcal{T}(\mathcal{C}^x)$. Then, for each $(d-1)$-concept $X \in \mathcal{T}(\mathcal{C}^x)$, we build the set X_i needed to extend X into a d-concept. An element y is added to X_i when $y \times \prod_{j \neq i} X_j \subseteq \mathcal{R}$, that is if there exists a $(d-1)$-dimensional box full of crosses (but not necessarily maximal) in $\mathcal{R}$, at level y. The final set X_i always contains at least x. To compute the set introducer concepts for a d-context, one needs to call Algorithm 3 on each dimension of a d-context. In some applications, it may be useful to compute the introducer concepts with respect to a given order $\lesssim_i$.

```
Algorithm 3: IntroducerDim($\mathcal{C}, i$)
Input: $\mathcal{C}$ a $d$-context, $i \in \{1, \dots, d\}$ a dimension
Output: $\mathcal{I}(\mathcal{S}_i)$ the set of introducer concepts of elements of dimension $i$
$I \leftarrow \emptyset$
foreach $x \in \mathcal{S}_i$ do
    $C \leftarrow \emptyset$
    foreach $X = (X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_d) \in \mathcal{T}(\mathcal{C}^x)$ do
        $X_i \leftarrow \emptyset$
        foreach $y \in \mathcal{S}_i$ do
            if $\prod_{j \neq i} X_j \times y \subseteq \mathcal{R}$ then
                $X_i \leftarrow X_i \cup y$
        $C \leftarrow C \cup (X_1, \dots, X_i, \dots, X_d)$
    $I \leftarrow I \cup C$
return $I$
```

5.2.1 A small combinatorics intuition: powerset d-lattices

Unlike the 2-dimensional case, where introducers are unique and the size of the AOC-poset is thus bounded by $|\mathcal{S}_1| + |\mathcal{S}_2|$, in the general case, it is bounded by $\mathbb{K}_{d-1} \times \sum_{i \in \{1,\dots,d\}} |\mathcal{S}_i|$, with $\mathbb{K}_d$ the maximal number of d-concepts in a d-context. Since a 1-context has only one 1-concept, this bound is reached in the 2-dimensional case.

Let us consider a powerset 3-lattice $\underline{\mathfrak{T}}(5)$ on a ground set of size 5. It is (well) known[1] that $\underline{\mathfrak{T}}(5)$ has $3^5 = 243$ concepts. However, the size of the introducer set of the powerset trilattice $\underline{\mathfrak{T}}(5)$ is 30. Indeed, by Proposition 64 we know that there exists a mapping between the 2-concepts of each layer induced by fixing an element of a dimension and the introducers. As, by definition, every layer of the context inducing a powerset trilattice has two 2-concepts, the number of introducer concepts in a powerset 3-lattice on a ground set of five elements is then bounded by $3 \times 5 \times 2$. This number is reached as the unique "hole" in each layer intersects all the concepts of the other layers. A more formal proof is given for a more general proposition below (Proposition 65).

In fact, for any powerset 3-lattice on a ground set of size n, the corresponding introducer d-ordered set has $3 \times 2 \times n$ elements. Figure 5.4 shows a 4-adic contranominal scale on three elements. Figure 5.5 shows the introducers of the powerset 3-lattice on a ground set of 3 elements.

Proposition 65. *Let d be an integer. A powerset d-lattice $\mathfrak{T}^d(n)$ on a ground set of n elements has d^n elements. Its corresponding introducer d-ordered set has $d \times (d-1) \times n$ elements.*

Proof. Let $\mathcal{C}$ be a d-dimensional contranominal scale, that gives rise to $\mathfrak{T}^d(n)$. Let x be an element of a dimension. The $(d-1)$-context $\mathcal{C}^x$ has only one hole (by definition of a contranominal scale). This implies that the complementary hypergraph for $\mathcal{C}^x$ has

[1] not that well known, but it is said in this paper [45]

		a	b	c	a	b	c	a	b	c
A	α		×	×	×	×	×	×	×	×
	β	×	×	×	×	×	×	×	×	×
	γ	×	×	×	×	×	×	×	×	×
B	α	×	×	×	×	×	×	×	×	×
	β	×	×	×	×		×	×	×	×
	γ	×	×	×	×	×	×	×	×	×
C	α	×	×	×	×	×	×	×	×	×
	β	×	×	×	×	×	×	×	×	×
	γ	×	×	×	×	×	×	×	×	
			1			2			3	

Figure 5.4: This is a 4-adic contranominal scale where the empty cells have been framed. Every layer induced by fixing an element (for example A) has three 3-concepts (in $\mathcal{C}^A$ we have $(123, \alpha\beta\gamma, bc)$, $(123, \beta\gamma, abc)$ and $(23, \alpha\beta\gamma, abc)$).

only one $(d-1)$-edge, and thus $d-1$ minimal transversals. This implies that each $(d-1)$-layer induced by fixing an element of a dimension has $d-1$ concepts. By Proposition 64, we know that there exists a mapping between the $(d-1)$-concepts of the layers and the introducer concepts of the context. Since we have d dimensions and n layers by dimension, the number of introducer concepts is bounded by $d \times (d-1) \times n$

Moreover, let $X = (X_1, \ldots, X_d)$ be an introducer concept for element x of dimension i. Then $X_i = x$. Indeed, by definition of a contranominal scale, there will be a 'hole' per layer of the context $\mathcal{C}$ that will be in the width of X.

This ensures that the $d \times (d-1) \times n$ introducer concepts that arise from Proposition 64 are distinct and that this number is reached. □

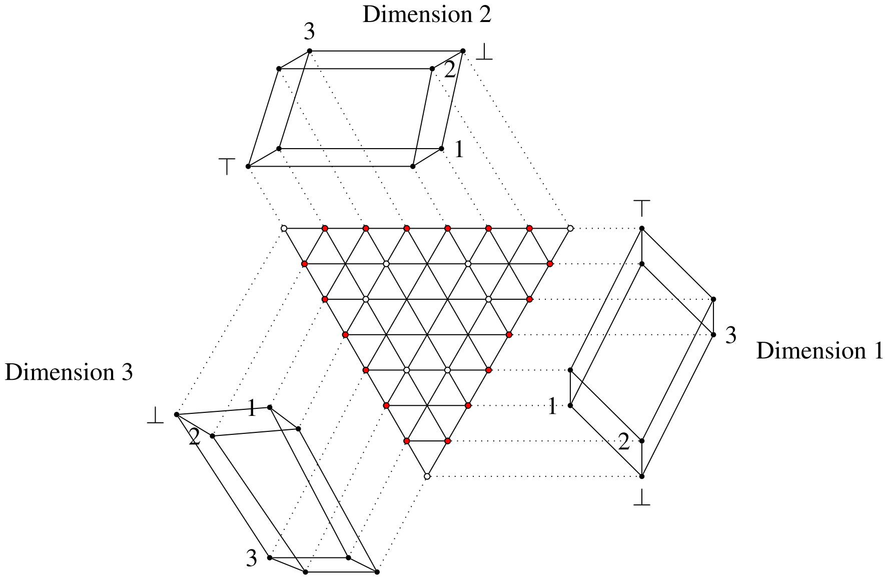


Figure 5.5: This represents a powerset 3-lattice. Its introducer concepts are filled in red.

Part III

Applications

Even in good conscience, when dealing with abstract objects such as d-lattices or hypergraphs, one can forget the strong relation that Formal Concept Analysis has with data. In this dissertation, we try to be as clear and mathematically rigorous as possible, and with these considerations, it is easy to stop thinking about how to use lattices in practice. We did not forget the data, nor did we stop thinking about uses and applications of lattices.

In this last part of the dissertation, we group three pieces of work into two chapters that can be labelled as more applied than the first part (Combinatorics) or the second (Algorithms). Here, we actually make use of Formal Concept Analysis and in a more general way of lattice theory in real life applications.

We still have two chapters to go! They pretty much have the same structure. First, we present the applicative context: "where does our problem come from?". Second, we present the proposed solution: "how do we address the problem?". Finally, we evaluate our solution: "did we propose a good solution?".

Chapter 6 presents a problem of exploratory search that takes its roots in software engineering. We introduce some algorithms to locally generate conceptual structures such as AOC-posets or some parts of a relational lattice family.

In Chapter 7, we present the context of implicit authentification, together with a classification method based on concept lattices, and some experiments. The authentification is implicit when you don't need to authenticate yourself, the system recognises you.

Chapter 6

On-demand object and (relational) attribute exploration

In this chapter we are interested in exploratory search. Exploratory search is an information retrieval strategy that aims at guiding a user in a space of documents to help them find their target. This process is adapted when a user is unfamiliar with the data space or when the data is too large to be known in its entirety.

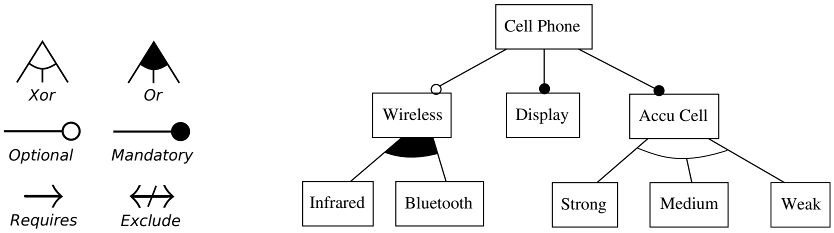
This chapter starts by a quick recall on exploratory search and a presentation of the applicative context that serves as a ground for our algorithms. Then, we outline algorithms to perform exploratory search in AOC-posets, through conceptual navigation. Finally, we change paradigm and leave AOC-posets in order to explore families of lattices under the formalism of Relational Concept Analysis.

6.1 Exploratory search

6.1.1 Applicative context

Software product line engineering (SPLE) [46] is a development paradigm which goal is to efficiently create and manage a collection of related software systems, based on their common characteristics.

A central point of SPLE is the modeling of the common parts and the variants contained in the related software systems, called the *variability* of the software product line. This variability is represented by variability models, which are the traditional starting points of information retrieval operations in software product lines, including product selection. Product selection is an important task that consists in guiding the user into selecting the functionalities they want in the final derived software system. The most prevalent approach to model variability relies on *feature models* (FMs) [47], a family of visual languages that describe a set of features and dependencies between them. Figure 6.1 depicts a feature model representing a software product line about cell phones. A combination of features respecting all the constraints expressed in the FM is called a *valid configuration*, and corresponds to a derivable software system.



$$Bluetooth \rightarrow Strong\ ;\ Infrared \nleftrightarrow Weak$$

Figure 6.1: Example of a feature model representing a software product line about Cell Phones

Current approaches for product selection rely on the feature models to automatically deploy configurations. These methods do not allow the user to change their final configuration without having to start the product selection from scratch, or to see which other configurations are similar to theirs. This can be considered too stiff for an average user who may want to modify their choice. In this context, it is interesting to apply exploratory search in the context of product selection to complement these methods and offer a more flexible selection.

6.1.2 Conceptual navigation: exploratory search in FCA

Conceptual navigation allows a user to start from an existing or partial configuration, explore similar ones, and be informed on how they can select or deselect features to obtain other valid configurations. It is noteworthy that the number of valid configurations depicted by a feature model grows exponentially with its number of features (most notably when any combination of features gives a valid configuration). Reducing the complexity of the underlying conceptual structure is important in order to conceive applications that would work in this context.

Lattice structures were among the first structures used to support information retrieval processes [48], and their usage was later generalized to Formal Concept Analysis. The concept lattice offers a convenient structure to do exploratory search, where navigating from concept to concept by selecting or deselecting attributes emulates iterative modifications of the document descriptor. Exploratory search by conceptual navigation has been used in several applications, for instance querying web documents [49] or browsing a collection of images [50].

FCA-based exploratory search raises some problems, mainly because of the size (in terms of number of concepts) of the concept lattices. Moreover, a user can rapidly get disoriented while navigating in such a large and convoluted structure.

Several methods have been proposed through the literature to reduce the complexity of conceptual navigation. In [48], the authors choose not to show the whole concept lattice to the user, but only a part of it, restricted to a focus concept and its neighborhood. This is the same navigation approach that we apply in this chapter, both on AOC-poset and on a relational lattice family. Other examples of uses can be found in several works [50–52].

In [53], the authors propose two methods to extract trees from concept lattices and use them as less complex structures for browsing and visualization. The difference between the two methods lies in the way the “best” parent for each concept in the tree is assigned: the first one is based on the selection of one parent per layer, and the second one on conceptual indexes. They then simplify again the final structure by applying two reduction methods based on fault-tolerance and clustering on the extracted trees. In [54], the authors propose a tool to build and visualize formal concept trees.

In [55], the authors give a FCA-based approach to select services and compose meaningful applications in a context of smart house. They aim at ensuring that, at runtime, the application is as adapted as possible to the environment. To this end, they use an approach that generates only part of the concept lattice (the filter of some starting configuration).

Carpineto and Romano [49] allow a user to bound the information space by dynamically applying constraints during the search to prune the concept lattice. Bounding allows to reduce the explorable data space and help the user focus on the parts they are interested in. Following the same idea, iceberg concept lattices [56] are pruned structures that only show the top-part of concept lattices which can be used to perform conceptual navigation, when some monotonous metric is involved.

In [52], the authors present `SearchSleuth`, a tool for local analysis of web queries based on FCA, that derives a concept and its neighborhood from a query. Because the domain cannot be computed entirely, it generates a new formal context at each navigation step: for each user query, it retrieves the list of results, extracts the relevant terms, and builds a context from these terms and their associated documents. The navigation is managed through an interface which suggests terms, making the underlying graph structure and its complexity implicit.

Alam et al. [51] present a tool, `LatViz`, that provides several operations to reduce the information space. One of them facilitates the visualization and the navigation. The authors propose to display the concept lattice *level-wise*: selecting a concept at a level ℓ displays all its sub-concept at level $\ell - 1$. Another functionality allows to prune the concept lattice by restricting navigation in sub-concepts and/or super-concepts of concepts selected by the user, in the same way as in [49]. Also, the tool lets one compute AOC-posets to support conceptual navigation: the authors describe AOC-posets as the “core” of their corresponding concept lattices. However, they compute the whole structure using the Hermes algorithm and do not propose an on-demand generation.

Greene el al. [57] discuss refinement and enlargement (broadening) approaches which are not restricted to neighbor concepts, and therefore allow navigation by non-minimal steps to ease exploratory search in large information spaces.

Exploratory search also exists in the context of association rules. This is closely related to the exploration in concept lattices. The study in [58] use exploration in a set of assocation rules. They introduce some reduction operators that allow to prune the set of rules to consider.

6.2 Conceptual navigation in AOC-posets

In order to jump from software engineering to Formal Concept Analysis, it is necessary to define the model that we will use. We need to define a set of objects, attributes and a relation between them. We will simply use the set of features as attributes. The set of objects is the set of valid configurations. The relation is straightforwardly derived from which valid configurations has which features. This application uses exclusively 2-dimensional contexts and 2-dimensional introducer posets.

6.2.1 Are AOC-posets good supports for exploratory search?

We recall that a concept introducing at least one attribute is called an *attribute-concept* (AC), and a concept introducing at least one object is called an *object-concept* (OC). A concept can introduce both an attribute and an object (*attribute-object-concept* (AOC)), or it can introduce neither of them (*plain-concept*). When using the simplified representation of extents and intents, plain-concepts appear in the lattice as concepts with empty extents and intents.

	Cell Phone	**Wireless**	**Infrared**	**Bluetooth**	**Display**	**Accu Cell**	**Strong**	**Medium**	**Weak**
c_1	×				×	×	×		
c_2	×				×	×		×	
c_3	×				×	×			×
c_4	×	×	×		×	×	×		
c_5	×	×	×		×	×		×	
c_6	×	×		×	×	×	×		
c_7	×	×	×	×	×	×	×		

Figure 6.2: Formal context depicting the 7 configurations of the software product line about cell phones. The objects are names c_i because, in our context, they represent configurations.

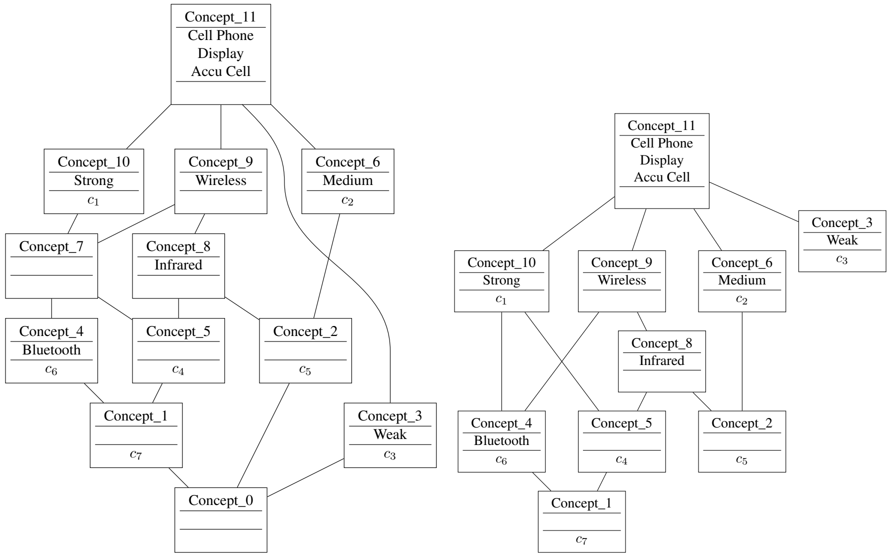


Figure 6.3: Concept lattice (left) and AOC-poset (right) associated with the formal context of Figure 6.2

Figure 6.2 shows the formal context associated with the feature model in Figure 6.1. In some types of applications, it is not necessary to take plain-concepts into account. For instance, this is the case when the lattice is only used as a support to organize objects by their attributes where they are therefore represented by their introducer concepts, and not to highlight maximal groups of elements. In these particular cases, one can benefit from only generating the sub-order restricted to the *introducer* concepts instead of the whole concept lattice. This smaller structure (in terms of number of concepts) is the *Attribute-Object-Concept partially ordered set* (AOC-poset).

Figure 6.3 (right) presents the AOC-poset associated with the context from Figure 6.2: it corresponds to the partial order of concepts from Figure 6.3 (left), minus Concept_0 and Concept_7. While the concept lattice associated to a formal context $(\mathcal{O}, \mathcal{A}, \mathcal{R})$ can have up to $2^{min(|\mathcal{A}|,|\mathcal{O}|)}$ concepts, the associated AOC-poset cannot exceed $|\mathcal{O}| + |\mathcal{A}|$ concepts.

AOC-posets can be used as a smaller alternative to concept lattices to structure a collection of objects depending on the attributes they share and navigate through this collection by selecting and deselecting attributes. However, when concept lattices represent all the possible queries that a user can formulate, AOC-posets restrict this set to the minimal queries required to perform conceptual navigation. As we have seen before, neighbor concepts in a concept lattice represent minimal possible modifications a user can make to the current query and therefore offer a data space in which one can navigate in minimal steps. This means that concept lattices allow to select and deselect non-cooccurrent attributes one by one. AOC-posets do not preserve the minimal step query refinement/enlargement property, but factorize the possible query modification steps to keep the most prevalent ones (the ones that correspond to *valid* *configurations*, or, in our model, objects). For instance, in the concept lattice of Figure 6.3 (left), if a user has selected Concept_4 as the current concept, they can choose to deselect attribute *Bluetooth* and thus move to Concept_7. From this concept, they can now choose to deselect either *Strong* or *Wireless* and move respectively to Concept_9 or Concept_10. In AOC-posets, because plain-concepts, that play the role of "transition steps", are not present, the choices that allow to move from concept to concept are condensed. This time, in Figure 6.3 (right), if a user wants to enlarge their query from Concept_4, they can either deselect both *Bluetooth* and *Strong* in one step to move to Concept_9, or deselect both *Bluetooth* and *Wireless* to reach Concept_10.

6.2.2 Exploration algorithm

On-demand, or local, generation consists in generating only the part of the structure we are interested in, and has already been applied to concept lattices, with algorithms such as NEXTCLOSURE [3]. To our knowledge, several algorithms exist to build AOC-posets: Ares, Ceres, Pluton and Hermes [59]. However, none of them performs on-demand generation of AOC-posets. In what follows, we outline algorithms to retrieve the conceptual neighborhood of a given concept in an AOC-poset.

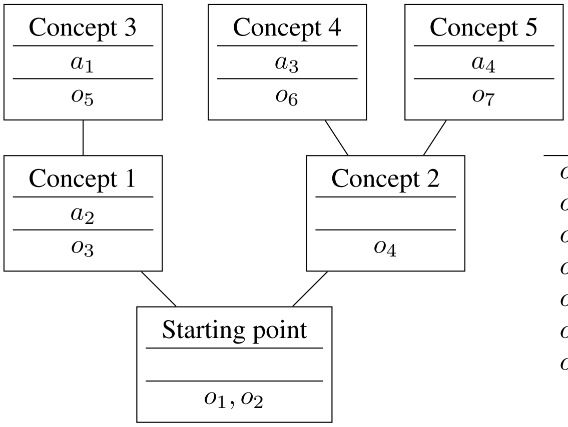
The exploration can start from the top concept (the most general query) when the user wants to make a software configuration from scratch. It is also possible that the user already has partial knowledge of their goal. In this case, it is necessary to be able to start the exploration from any set of features. As the concept corresponding to the (potentially partial) configuration that the user has in mind does not necessarily introduce an object or an attribute, we suppose that the input of the exploration is a formal concept, plain or not. The problem is thus to compute, for any concept in a concept lattice, its conceptual neighborhood in the AOC-poset.[1]

Let us start with computing the upper covers. We are looking for the smallest attribute-concepts or object-concepts greater than the input. We start out by computing the smallest attribute-concepts greater than our input concept C. They can be obtained by computing the concepts $(\{a\}', \{a\}'')$ for each attribute a in the intent of C. We remark that a concept $(\{a_1\}', \{a_1\}'')$ is a super-concept of another concept $(\{a_2\}', \{a_2\}'')$ if and only if a_1 is in $\{a_2\}''$. As such, the smallest attribute-concepts are the ones that are computed from attributes that do not appear in the closures of other attributes.

Once we have the smallest attribute-concepts, we want to compute the smallest object-concepts that are between them and C. This means that we are looking for concepts of the form $(\{o\}'', \{o\}')$ such that o is in the extent of one of the attribute concepts we have and $\{o\}'$ is contained in the intent of C. We remark once again that a concept $(\{o_1\}', \{o_1\}'')$ is a super-concept of another concept $(\{o_2\}', \{o_2\}'')$ implies o_2 is in $\{o_1\}''$ and that the closures of some objects give us information on object-concepts that cannot be minimal.

An illustration of this process is shown in Figure 6.4, that shows a context and its associated AOC-poset (in the lightened representation).

[1] If the concept is an introducer, then we need its conceptual neighborhood in the AOC-poset. If the starting concept is not an introducer and as such is not in the AOC-poset, we want its closest neighbors that are introducers.



	a_1	a_2	a_3	a_4
o_1	×	×	×	×
o_2	×	×	×	×
o_3	×	×		
o_4			×	×
o_5	×			
o_6			×	
o_7				×

Figure 6.4: First, we compute the attribute-concepts corresponding to the attributes in the intent of our starting concept. This gives us concepts 1, 3, 4 and 5. Then, by keeping only the concepts that are computed from attributes that do not appear in the closures of other attributes, we are left with concepts 1, 4 and 5. The next step is to compute the object-concepts that are between those attributes-concepts and the starting concept. In this example, we obtain concept 2.

Proposition 66. *Algorithm 4 computes the upper cover of the input concept in the AOC-poset.*

The first loop computes the closure of single attributes. Each closure allows us to remove attributes that correspond to non-minimal attribute-concepts. The resulting set R contains the intents of the attribute-concepts that are both super-concepts of C_i and minimal for this property.

The second loop constructs the set O of objects that are in the extent of an element of R but not in the extent of C_i.

The third loop removes the objects of O that cannot possibly be introduced by a superset of C_i.

Finally, the fourth loop removes the objects of O that produce non-minimal object-concepts. The attribute-concepts that are no longer minimal are also removed.

Therefore, considering the initial configuration, the object-concepts introduce the most similar and more generalized configurations, and the attribute-concepts show the factorized possible attribute deselections the user can make.

Computing the lower covers is done using the same algorithm, exchanging the roles of attributes and objects. This time, object-concepts present the most similar and more specialized configurations, and the attribute-concepts the possible condensed attribute selections.

Algorithm 4: Upper Cover

```
Input: A concept $C$
Output: The upper covers of $C$ in the AOC-poset
$A \leftarrow Int(C)$
foreach $a \in A$ do
    $Y \leftarrow \{a\}''$
    $A \leftarrow A \setminus \{Y \setminus \{a\}\}$
$R \leftarrow \{\{a\}'' | a \in A\}$
$O \leftarrow \emptyset$
forall $S \in R$ do
    $X \leftarrow S'$
    $O \leftarrow O \cup (X \setminus Ext(C))$
forall $o \in O$ do
    if $o' \not\subset Int(C)$ then
        $O \leftarrow O \setminus \{o\}$
forall $o \in O$ do
    $T = \{S | (S \in R) \wedge (o \in S')\}$
    $R \leftarrow R \setminus T$
    $Y \leftarrow \{o\}''$
    if $\exists p \in O$ such that $p \in Y$ then
        $O \leftarrow O \setminus \{o\}$
$R \leftarrow \{(\{o\}'', \{o\}') \mid o \in O\}$
return $R$
```

6.2.3 Evaluation

We have implemented our algorithms, and we tested them on software product line datasets extracted from the SPLOT repository[2]. SPLOT (for *Software Product Line Online Tools*) is an academic website providing a repository of feature models along with a set of tools to create, edit and perform automated analysis on them. We have selected thirteen representative feature models which describe software product lines as e-shops, cell phones or video games, from small sizes (13 configurations) to larger ones (4774 configurations). To test our method on data extracted from feature models, we first create a formal context $configurations \times features$ for each one of them. Then, from a context, our implementation allows to find a concept corresponding to a subset of features and compute its conceptual neighborhood in AOC-posets.

In this experiment, we assume that a user will not exceed 50 navigation steps, as they want to be familiarized to the similar valid configurations around their initial selection of features. To measure the gain of our method in terms of number of computed concepts, we compare for each context the number of computed concepts for 50 navigation steps (i.e., a concept and its neighborhood) to the total number of concepts in the associated AOC-poset and concept lattice. The results are presented in Fig-

[2] http://www.splot-research.org/

ure 6.5. The concept lattice curve represents an upper-bound to visualize more easily the gain of AOC-posets and local-generation of AOC-posets comparing to concept lattices.

Figure 6.5 shows that both aspects of our method (the use of AOC-posets and their partial generation) are useful for datasets that generate lattices with a size around seven hundred concepts or more. The difference between AOC-posets and concept lattices when the structures are small is not very important (e.g., 19 concepts against 25, 131 against 166), but AOC-posets become very interesting with larger structures (e.g., 1074 concepts against 5761, 669 against 6430).

For small datasets, 50 navigation steps create some redundancy and thus a big number of computed concepts. For bigger datasets, 50 navigation steps allow to visit an important number of concepts (several hundreds), while still being small with regards to the whole concept lattice.

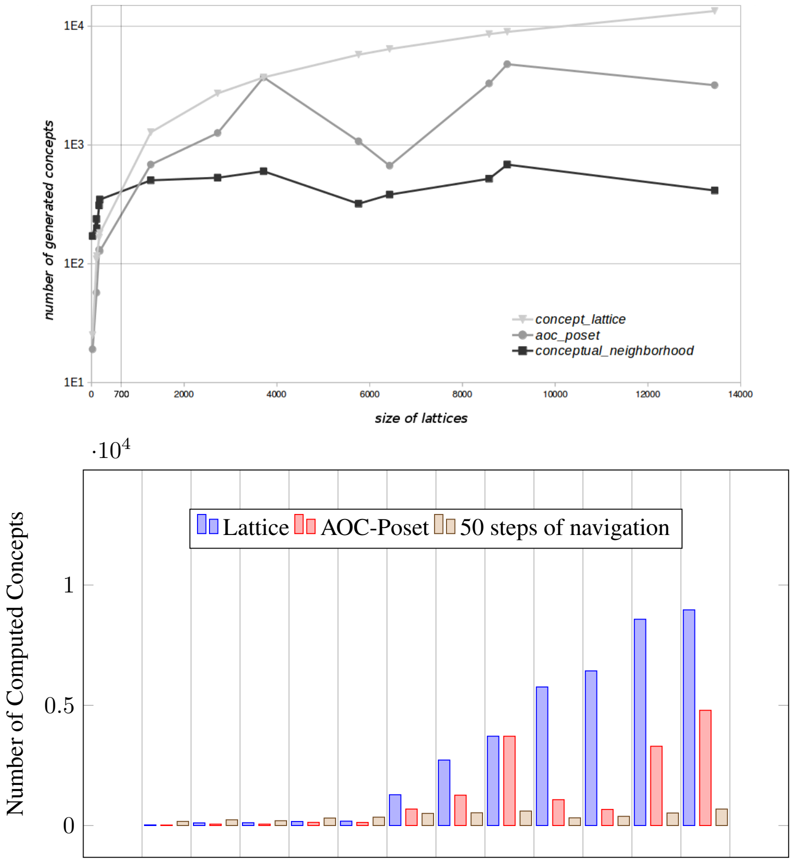


Figure 6.5: Number of generated concepts for AOC-posets and conceptual neighborhood for 50 navigation steps, depending on the size of their associated concept lattices (logarithmic scale)

We also present some further experiments on some other datasets. We consider nineteen datasets for which the size of the lattices range from 298 to 4361. On the next figure, Figure 6.6, we represent, for each dataset, the number of computed concepts in the concept lattice, and then for some numbers of steps depending on the size of the lattice or the number of objects (square root or logarithmic value). We show the number of concepts generated for a hundred steps as an indication.

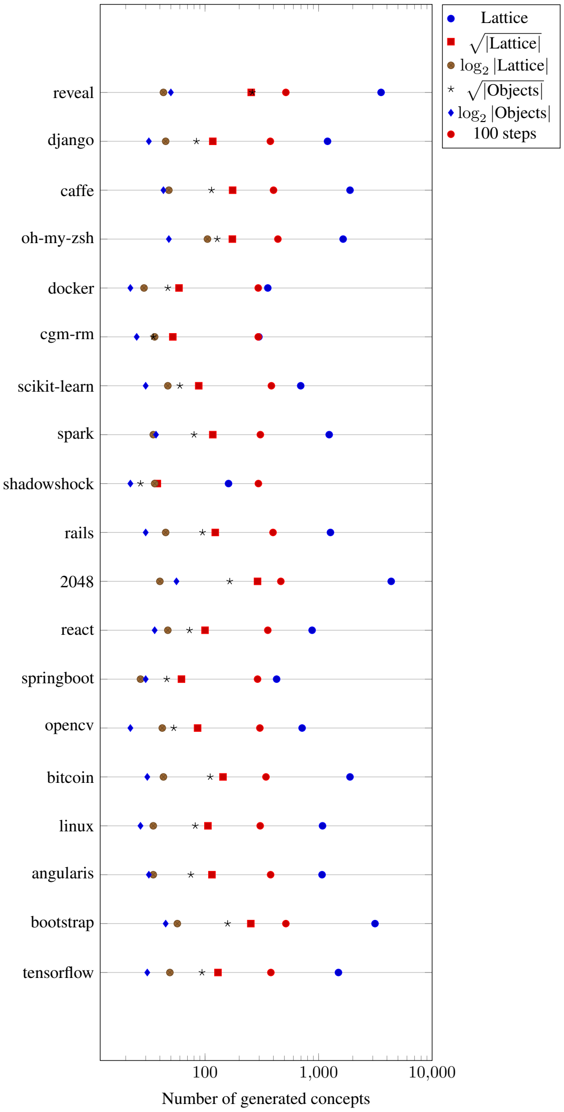


Figure 6.6: Each line corresponds to a dataset. We consider that the number of configurations that a user wants to explore can be a function of the number of valid configurations or the number of concepts in the lattice.

6.3 Conceptual navigation and RCA

In this section, we continue the work of exploratory search on a new setting. Instead of looking at concept lattices or AOC-posets, we are now interested in a Relational Concept Analysis (RCA) lattice family. The remainder of this chapter is organized as follows: we start by introducing RCA and the principal task that we want to address that is conceptual navigation in a family of lattices. Then, we introduce some algorithms that allow to locally generate a relational family. The last section shows an example of a step of our algorithms.

6.3.1 Notions of Relational Concept Analysis

Relational Concept Analysis (RCA) [60–62] is an extension of Formal Concept Analysis to multirelational data. Instead of using only a context as starting point, the data structure on which RCA is based is a *relational context family*: a family of contexts and relations between the objects of those contexts.

Definition 67. *A* relational context family *is a pair* $(\mathbf{K}, \mathbf{R})$ *where*

- $\mathbf{K}$ *is a set of formal contexts* $K_i = \{\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i\}$*, with* $\mathcal{O}_i$ *the set of objects of context* K_i*,* $\mathcal{A}_i$ *its set of attributes and* $\mathcal{I}_i$ *the relation between them;*

- $\mathbf{R}$ *is a set of relational contexts* $(\mathcal{O}_i, \mathcal{O}_j, r_{ij})$*, where* $\mathcal{O}_i$ *is the set of objects of context* K_i *from* $\mathbf{K}$*,* $\mathcal{O}_j$ *is the set of objects of context* K_j *from* $\mathbf{K}$ *and* $r_{ij} \subseteq \mathcal{O}_i \times \mathcal{O}_j$ *is a relation between them. Context* K_i *is called the source context of the relation while context* K_j *is called the target context of the relation.*

For an object o of formal context K_i and a relation r_{ij} between K_i and K_j, we denote by $r_{ij}(o)$ the set of objects from K_j that are related to o by r_{ij}.

Tables 6.1 and 6.2 show an example of a relational context family $(\mathbf{K}, \mathbf{R})$. This relational context family is borrowed from an application of software product line engineering. Table 6.1 shows the contexts of $\mathbf{K}$ while Table 6.2 shows the relational context of $\mathbf{R}$. In $\mathbf{K}$, the table labeled *DM_tools* shows five data modeling tools, with their compatible operating systems (OS) and the data models (DM) that they support. The table labeled *DBMS* describes four database management systems and the data types (DT) that they can handle.

In $\mathbf{R}$, the table *Supports* represents a relational context that states which data modeling tools support which database management systems. This context represents a relation between the objects of *DM_tools* and those of *DBMS*.

RCA works by iterative steps. The first step will build one concept lattice per context in $\mathbf{K}$ (the concept lattices associated with the two contexts of $\mathbf{K}$ are shown in Figure 6.7). The links that are expressed in $\mathbf{R}$ are not taken into account at this point.

The links are then introduced between objects using *relational attributes*: they introduce the concepts of the target context in the source context through a relation and a *scaling operator*. The relational attributes are integrated in the contexts as new attributes. A context K augmented with new relational attributes is usually denoted K^+. A relational attribute has the form "$\rho\ r.C$" where ρ is a scaling operator, r is a relation and C is a concept from the target context, that has some properties with

$\mathbf{K} =$

DM_tools	OS:Windows	OS:Mac OS	OS:Linux	DM:Conceptual	DM:Physical	DM:Logical	DM:ETL
Astah	×	×	×	×			
Erwin DM	×			×	×	×	
ER/Studio	×			×	×	×	×
Magic Draw	×	×	×	×	×	×	
MySQL Workbench	×	×	×		×		

DBMS	DT:Enum	DT:Set	DT:Geometry	DT:Spatial	DT:Audio	DT:Image	DT:Video	DT:XML	DT:JSON	DT:Period
MySQL	×	×	×							
Oracle				×	×	×	×	×		
PostgreSQL	×		×					×	×	
Teradata	×		×					×	×	×

Table 6.1: Two formal contexts: (above) Data Modelling tools (*DM_tools*) and (below) DataBase Management Systems (*DBMS*).

$\mathbf{R} =$

Supports	MySQL	Oracle	PostgreSQL	Teradata
Astah	×	×		
Erwin DM	×	×		×
ER/Studio	×	×	×	×
Magic Draw	×	×	×	
MySQL Workbench	×			

Table 6.2: This relation context shows a relation between the set of objects of both contexts from $\mathbf{K}$.

respect to the relation and the scaling operator. We will soon give examples of some links that arise from our example. First, we need to define the scaling operators and how they allow to represent those links.

We focus on two scaling operators: the *existential* scaling operator $\exists$ and the *universal strict* scaling operator $\exists\forall$.

Definition 68 (Existential Scaling). *Let K_i and K_j be formal contexts of* $\mathbf{K}$. *Let r_{ij} be a relation between them. For every object o of K_i and every concept C of the target context K_j, if $r_{ij}(o)$* ***intersects*** *the extent of C, then we extend the set of attributes of K_i with the relational attribute $\exists\ r_{ij}.C$ and add it to the attributes that describe o in K_i.*

Let us consider the relation *Supports* from Table 6.2. We apply the existential scaling between *DM_tools* and *DBMS*. For each object o of *DM_tools*, we look at which concepts of *DBMS* have some object in their extent that is related to o by the relation *Supports*. For those concepts, we add a relational attribute in *DM_tools* that contains the scaling operator $\exists$, the name of the relations (abbreviated "sup") and the identifier of the concept. The result of the existential scaling on *DM_tools* is shown in Table 6.3.

Definition 69 (Universal strict scaling). *Let K_i and K_j be formal contexts of* $\mathbf{K}$. *Let r_{ij} be a relation between them. For every object o of K_i and every concept C of the*

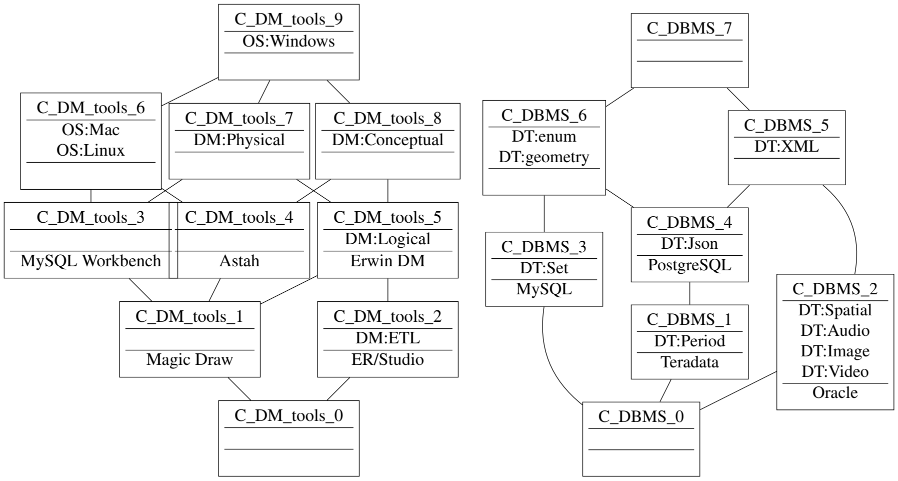


Figure 6.7: Concept lattices associated with *DM_tools* (left) and with *DBMS* (right).

DM_tools$^+$	OS:Windows	OS:Mac OS	OS:Linux	DM:Conceptual	DM:Physical	DM:Logical	DM:ETL	∃ sup.C_DBMS_0	∃ sup.C_DBMS_1	∃ sup.C_DBMS_2	∃ sup.C_DBMS_3	∃ sup.C_DBMS_4	∃ sup.C_DBMS_5	∃ sup.C_DBMS_6	∃ sup.C_DBMS_7
Astah	×	×	×	×						×	×		×	×	×
Erwin DM	×			×	×	×			×	×	×	×	×	×	×
ER/Studio	×			×	×	×	×		×	×	×	×	×	×	×
Magic Draw	×	×	×	×	×	×				×	×	×	×	×	×
MySQL Workbench	×	×	×		×						×			×	×

Table 6.3: Formal context *DM_tools* extended according to the relation *Support*, with the scaling operator ∃.

target context K_j, if $r_{ij}(o)$ ***is a non-empty subset*** *of the extent of C, then we extend the set of attributes of K_i with the relational attribute $\exists\forall\ r_{ij}.C$ and add it to the attributes that describe o in K_i.*

Let us consider the relation *Supports* from Table 6.2. We apply the universal strict scaling between *DM_tools* an *DBMS*. For each object o of *DM_tools*, we look at which concepts of *DBMS* contain all the relatives of o by r_{ij} in their extent. For those concepts, we add a relational attribute in *DM_tools* that contains the scaling operator ∃∀, the name of the relations (abbreviated "sup") and the identifier of the concept. The result of the universal strict scaling on *DM_tools* is shown in Table 6.4.

After applying a scaling operator, the relational context family is updated: the contexts from $\mathbf{K}$ may have been extended with new relational attributes. We call this new set of contexts $\mathbf{K}_s$, for step s. The concept lattice of those extended contexts structures the objects by their attributes and their relational attributes, allowing the links to be represented.

DM_tools$^+$	OS:Windows	OS:Mac OS	OS:Linux	DM:Conceptual	DM:Physical	DM:Logical	DM:ETL	∃∀ sup.C_DBMS_0	∃∀ sup.C_DBMS_1	∃∀ sup.C_DBMS_2	∃∀ sup.C_DBMS_3	∃∀ sup.C_DBMS_4	∃∀ sup.C_DBMS_5	∃∀ sup.C_DBMS_6	∃∀ sup.C_DBMS_7
Astah	×	×	×	×											×
Erwin DM	×			×	×	×									×
ER/Studio	×			×	×	×	×								×
Magic Draw	×	×	×	×	×	×									×
MySQL Workbench	×	×	×		×						×			×	×

Table 6.4: Formal context *DM_tools* extended according to the relation *Support*, with the scaling operator ∃∀.

The succession of steps as described above allows to reach some further relations: in a more complex data model that includes more than one relation, RCA will produce a succession of concept lattices that will be extended at each step with the new relational attributes. At step 0, the concept lattices are simply the ones built from the initial formal contexts, as shown in Figure 6.7. At step s, the formal contexts from $\mathbf{K}_s$ are extended with relational attributes that depend on the concepts of the concept lattices of $\mathbf{K}_{s-1}$ and the relations expressed in $\mathbf{R}$. The process stops either when a maximal depth or a fix point[3] is reached. In the process of RCA as described here, a fix point exists and is garanteed to be reached: in [62], Mohamed Rouane Hacene, Marianne Huchard, Amedeo Napoli and Petko Valtchev give a sequence that describes the process of extention of the contexts, and show that it stops by observing that the number of objects does not change, hence the process is bounded by the Boolean lattice of the set of objects.

Tables 6.3 and 6.4 are good indicators that a context K_i might be extended by a lot of relational attributes (the size of the lattice associated to the target context of the relations starting from K_i). In order to reduce this growth, a variant of RCA using AOC-posets has been introduced: RCA-AOC [63]. The rules of RCA-AOC are almost the same as the rules of RCA: instead of choosing the concept that satisfy the rule associated with a scaling operator in the concept lattice, we select them in the AOC-poset.

The principal difference between classical RCA and RCA-AOC is that RCA-AOC does not always reach a fix point.[4] In our application, we tackle this problem by setting a maximal depth for our process.

6.3.2 Conceptual navigation in a relational context family

Let us describe the process that we want to set up in order to navigate in a relational context family. Our starting point is a concept C of a context K_i. From now on, K_i will be called our starting context and C our starting concept. The first thing that we

[3] A fix point is reached when the contexts from a step are identical to the contexts from the previous step and the lattice family is isomorphic to that of the previous step.

[4] There is no official publication where this is proved, but an example exists, if I'm to believe Marianne Huchard's word (and I do).

have to define is our exploration *strategy*. An exploration strategy is a set of pairs of the form (r_{ij}, ρ) where the r_{ij} are relations between the starting context and another context K_j of the RCF and the ρ are scaling operators. The relational neighborhood of a concept is the set that consists of its upper cover, lower cover and related concepts in other lattices. That is the output that we want to be able to visualize from C. In fact, the real output of our exploration algorithm is the *closed* relational neighborhood, that is the relational neighborhood together with a completed version of C with relational attributes. Since our process may consist of various of the above-described steps, the relational context family is also updated with relational attributes.

This process allows a user to choose which relations they want to follow in their exploration (by the way of the strategy). At each step, we update part of the relational context family (contrasting with classical RCA, where everything is updated at each step). We cannot guarantee that the process converges. In order to tackle that problem, we also consider that the user defines a maximal depth for their exploration.

6.3.3 Redefining the derivation operators

The explicit knowledge of all the relational attributes of a context requires the computation of all the concepts of all its related contexts. In an exploration setting, we don't want to spare the time that amounts to the exhaustive computation of the relational concepts of multiple contexts. We would prefer to manipulate only a minimal number of relational attributes that would allow us to derive, on-the-fly, the other relational attributes.

From now on, we use the notation $a \sim \rho\, r.(X, Y)$ to indicate that a is a relational attribute that uses ρ as a scaling operator, a relation r and a concept (X, Y).

Any object described by a relational attribute $\rho\, r.(X, Y)$ is also necessarily described by all the relational attributes associated to more general concepts, that is attributes of the form $\rho\, r.(X_2, Y_2)$ with Y_2 a subset of Y. Intents can thus be represented without loss of information by their relational attributes constructed from attribute-wise maximal concepts (we will refer to them as maximal relational attributes for short in the sequel). However, a problem arises with such representation: the set intersection cannot be used to compute the intent of a set of objects anymore. Similarly, if only maximal relational attributes are explicitly present in the context, the extent of a set of attributes cannot be computed through a simple test of set inclusion. To remedy this problem, we provide three algorithms to be used in place of the traditional derivation operators on set of objects or attributes (both classical or relational) when only the maximal relational attributes are given explicitly.

Algorithm 5 INTER takes as input two sets of attributes A and B represented by their maximal relational attributes. It outputs the set of maximal relational attributes of their intersection. A relational attribute $\exists\, r.(X_1, Y_1)$ is in the intersection of A and B if and only if there exists an attribute $\exists\, r.(X_2, Y_2)$ in A and an attribute $\exists\, r.(X_3, Y_3)$ in B such that X_1 is included both in X_2 and in X_3. The same holds for the $\exists\forall$ operator. Algorithm 5 computes the maximal relational attributes of the intersection of A and B by recursively intersecting the intents of the concepts used to build the relational attributes in A and B. The fact that, at any given time, the depth of all the relational attributes is bounded ensures that the recursion ends.

Algorithm 5: $\textsc{Inter}(K_i, A, B)$

```
Input: $K_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $A \subseteq \mathcal{A}_i$ a set of attributes,
       $B \subseteq \mathcal{A}_i$ the intent of an object $o$.
Output: The relational intersection of the set $A$ and the intent of $o$.
$R \leftarrow A \cap B$
$\mathcal{F} \leftarrow \emptyset$
foreach $a \sim \exists\, r_{ij}.(X_1, Y_1) \in B$ do
    foreach $a_2 \sim \exists\, r.(X_2, Y_2) \in A$ do
        $\mathcal{F} \leftarrow \mathcal{F} \cup \{\exists\, r.(\textsc{Ex}(K_j, \textsc{Inter}(K_j, Y_1, Y_2)), \textsc{Inter}(K_j, Y_1, Y_2))\}$
$R \leftarrow R \cup \textsc{Max}(\mathcal{F}, \subseteq_{A_i})$
$\mathcal{F} \leftarrow \emptyset$
foreach $a_1 \sim \exists\forall\, r.(X_1, Y_1) \in B$ do
    foreach $a_2 \sim \exists\forall\, r.(X_2, Y_2) \in A$ do
        $\mathcal{F} \leftarrow \mathcal{F} \cup \{\exists\forall\, r.(\textsc{Ex}(K_j, \textsc{Inter}(K_j, Y_1, Y_2)), \textsc{Inter}(K_j, Y_1, Y_2))\}$
$R \leftarrow R \cup \mathcal{F}$
return $R$
```

$\mathcal{K}$	a	b	c	$\exists r.(X_1, Y_1)$	$\exists r.(X_2, Y_2)$
o_1	×		×		×
o_2	×	×		×	×
o_3		×	×	×	

Figure 6.8: This formal context has two relational attributes by relation r (the relation is not given here).

Let us illustrate INTER with an example. We consider the context in Figure 6.8. This context has some attributes and some relational attributes. Say that we want to intersect the attribute $\{a, \exists\ r.(X_1, Y_1)\}$ with the intent of object o_1. We call Algorithm 5 with parameters $\mathcal{K}, \{a, \exists\ r.(X_1, Y_1)\}, \{a, c, \exists\ r(X_2, Y_2)\}$. At the beginning, the set R gets the set intersection of both input sets, that is $\{a\}$. However, since the relational attributes are only known implicitly, it is possible that there are some relational attributes that are not maximal for the intent (and thus not explicitly given in our context) that are still shared by both sets of attributes. INTER addresses this problem by recursively computing the intents of the concepts that are used to build the relational attributes.

Since we are redefining the derivation operators, in the rest of this section we will not use $\cdot'$ as the classical derivation operator. Instead, we will use $Intent(\{o\})$ to denote the set of attributes (relational or not) that describe an object o in a context. This notation is used in Algorithm 6 (IN). Algorithm 6 uses INTER to compute the intent of a set of objects described only by their maximal relational attributes. It starts with the set of all explicitly known attributes and intersects it with the description of each object of the input. The descriptions of the objects, while they are known implicitly, are completed with INTER for each object.

Algorithm 6: $\textsc{In}(K_i, O)$

```
**Input:** $K_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $O \subseteq \mathcal{O}_i$ a set of objects.
**Output:** Computes the intent of a set of objects $O$.
$A \leftarrow \mathcal{A}_i$
foreach $o \in O$ do
    $A \leftarrow \textsc{Inter}(K_i, A, Intent(\{o\}))$
return $A$
```

Algorithm 7 (Ex) computes the extent of a set of maximal relational attribute A. For each object o of the context and each relational attribute $\rho\ r.(X, Y)$ in A, Ex checks whether $r(o)$ and X intersect in the correct way, depending on which scaling operator is used. When a is not a relational attribute, then Ex computes its extent in the classical way. Going back to our example in Figure 6.8, say we want to compute the extent of $\{a, b, \exists\ r.(X_1, Y_1)\}$. We start by taking all the objects of our context. Then, for each scaling operator, we check for each object if its image by the relation intersects the extent of the corresponding concept, in the correct way. If it does not, it is removed. At the end, we use the usual set intersection for non-relational attributes.

Algorithm 7: $\textsc{Ex}(K_i, A)$

```
**Input:** $K_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $A \subseteq \mathcal{A}_i$ a set of attributes.
**Output:** Computes the extent of a set of attributes $A$.
$O \leftarrow \mathcal{O}_i$
foreach $a \in A$ do
    if $a \sim \exists\forall\ r.(X, Y)$ then
        foreach $o \in O$ do
            if $r(o) \not\subseteq X$ then
                $O \leftarrow O \setminus o$
    else if $a \sim \exists\ r.(X, Y)$ then
        foreach $o \in O$ do
            if $r(o) \cap X = \emptyset$ then
                $O \leftarrow O \setminus o$
    else
        foreach $o \in O$ do
            if $(o, a) \notin \mathcal{I}_i$ then
                $O \leftarrow O \setminus o$
return $O$
```

6.3.4 Computing the closed relational neighborhood

Since we do not want only the close relational neighborhood of a concept at depth one, but at any depth, we also need to extend the starting context with new relational attributes. We do so in Algorithm 8 (GROWCONTEXT). Algorithm 8 takes as input a context, a relation from K_i to K_j, a scaling operator, an object o and the set of object-concepts of K_j. It constructs new relational attributes using the scaling operator, the relation and the object-concepts, adds them to the context K_i and completes the incidence relation accordingly. As the set of object-concepts contains the irreducible elements of the lattice, relational attributes constructed with them are enough to reconstruct all the other possible relational attributes.

Algorithm 8: GROWCONTEXT$(K_i, r_{ij}, \rho, o, OC_j)$

```
Input: $K_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$ a formal context, $r$ a relation from $K_i$ to $K_j$, $\rho$ a
       scaling operator, $o \in \mathcal{O}_i$ an object and $OC_j$ the set of object-concepts
       of $K_j = (\mathcal{O}_j, \mathcal{A}_j, \mathcal{I}_j)$.
Output: Extends the context $K_i$ and adds the crosses.
if $\rho == \exists$ then
    foreach $(X, Y) \in OC_j$ such that $r(o) \cap X \neq \emptyset$ do
        $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \{\exists\, r.(X, Y)\}$
        $\mathcal{I}_i \leftarrow \mathcal{I}_i \cup (o, \exists\, r(X, Y))$
if $\rho == \exists\forall$ then
    $Y \leftarrow \mathrm{IN}(K_i, r(o))$
    $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \{\exists\forall\, r(\mathrm{EX}(K_i, Y), Y)\}$
    $\mathcal{I}_i \leftarrow \mathcal{I}_i \cup (o, \exists\forall\, r(\mathrm{EX}(K_i, Y), Y))$
```

Now that we have redefined the derivation operators on implicitly known relational contexts and know how to extend a context with relational attributes, we are able to compute the upper, lower and relational covers of a concept.

Relational cover The easiest are the relational covers. Given a starting concept (U, V), a concept (X, Y) is its relational cover if and only if $\rho\, r.(X, Y)$ is a maximal relational attribute in V.

Upper cover Upper covers are relatively easy to compute too. Candidates can be generated by adding an object – the set of which we have perfect knowledge of at any step of the exploration – to the current extent and computing the corresponding concept. The cover is the set of candidates that have the smallest extent.

Lower cover The lower covers are more challenging. The concepts from the lower cover could be computed by adding attributes to the intent, but, since the full set of relational attributes is known only implicitly, this approach would not work. We chose to, instead, remove objects. The lower covers of a concept (U, V) are the concepts with the maximal extents that are contained in U and do not contain any of the minimal generators of U. A simple way to compute them would be to remove minimal transversals of the minimal generators of U.

Algorithm 9 computes the closed relational neighborhood of a concept C. It takes as input a set of formal context $\mathbf{K} = \{K_1, \dots K_w\}$ of a RCF, a strategy $\mathcal{S} = \{(r_{ij}, \rho), \dots\}$ with $j \in \{1, \dots, w\}$ and a starting concept C from a context K_i of $\mathbf{K}$. The goal is to compute (or complete) the intent corresponding to the extent of C (possibly augmented by new relational attributes), as well as its upper, lower and relational covers, in the extended context K_i^+.

For each pair (r_{ij}, ρ), the first loop (Lines 1 to 4) computes the objet-concepts of the target context. Then, for each of those object-concepts, we check whether, together with the relation r_{ij} and the scaling operator, it gives rise to a new relational attribute that is added to the context K_i in GROWCONTEXT.

In Line 5, the intent of concept C is extended with the relational attributes that were added to the context during the previous loop. The next loop (Lines 6 to 8) computes the relational cover $\mathcal{R}$ of concept C. For each relational attribute in the intent of C, the corresponding concept (in the target context) is added to the cover. In the end of the loop, all the relational cover of C is computed.

In Lines 9 to 12, the upper cover $\mathcal{U}$ of C is computed. Candidates are created by adding an object to the extent of C. Only the extent-wise minimal candidates are kept to be the upper cover of C.

Finally, in Lines 13 to 15, the lower cover $\mathcal{L}$ of C is computing by removing from the extent of C a minimal transversal of the set of its minimal generators.

6.3.5 Example of a step

In this section we give an example of a step based on the RCF given in Tables 6.1 and 6.2. We recall that $\mathbf{K} = \{\textit{DM\_tools}, \textit{DBMS}\}$ and $\mathbf{R} = \{\textit{Supports}\}$. The exploration strategy that we want to apply considers the relation *Supports* with the existential scaling operator. Thus, $\mathcal{S} = \{(\textit{Supports}, \exists)\}$. Let us start with the concept C_DM_tools_8 = $\{(\{\text{Astah, Erwin DM, ER/Studio, Magic Draw}\}, \{\text{OS:Windows, DM:Conceptual}\})\}$.

The first loop of Algorithm 9 goes through every pair of the strategy in order to extend the starting context with respect to new relational attributes. Our strategy consists in only one pair, with a relation that goes from *DM_tools* to *DBMS*. We start by computing the OC-poset of *DBMS*, that consists of four concepts shown in Figure 6.9.

For each object o of the starting context *DM_tools*, we call function GROWCONTEXT (Algorithm 8) with parameters *DM_tools*, *Supports*, $\exists$, o and the set of object-concepts of Figure 6.9. This algorithm checks which relational attributes we sould extend the starting context with, and add the crosses into the relation. Let us consider the object Astah. The domain of Astah through relation *Supports* is the set $\{\text{MySQL, Oracle}\}$. For each object-concept (X, Y) that intersects the domain of Astah, we add a relational attribute $\exists$ *Supports*.(X, Y) to the starting context. We also add the crosses between Astah and the relational attribute. When we have done that for all the objects of the starting context, we have the extended context shown in Table 6.5. After the update of the starting context, we update the intent of our starting

```
Algorithm 9: RCA($\mathbf{K}, \mathcal{S}, C, K_i$)
   Input: $\mathbf{K} = \{K_1, \dots, K_w\}$ a set of formal contexts,
          $\mathcal{S} = \{(r_{ij}, \rho), \dots\}, j \in \{1, \dots, w\}$ a strategy, $C = (O, A)$ a concept
          of context $K_i = (\mathcal{O}_i, \mathcal{A}_i, \mathcal{I}_i)$.
   Output: $C, \mathcal{U}, \mathcal{R}, \mathcal{L}$ the completed concept $C$ and its relational
           neighborhood.
 1 foreach $(r_{ij}, \rho) \in \mathcal{S}$ do
 2     $OC_j \leftarrow$ ObjectPoset($K_j$)
 3     foreach $o \in \mathcal{O}_i$ do
 4         GrowContext($K_i, r_{ij}, \rho, o, OC_j$)
 5 $A \leftarrow$ In($K_i, O$)
 6 $\mathcal{R} \leftarrow \emptyset$
 7 foreach $a \sim \rho\, r.(X_1, Y_1) \in A$ do
 8     $\mathcal{R} \leftarrow \mathcal{R} \cup \{(X_1, Y_1)\}$
 9 $\mathcal{U} \leftarrow \emptyset$
10 foreach $o \in \mathcal{O}_i \setminus O$ do
11     $\mathcal{U} \leftarrow \mathcal{U} \cup \{($Ex$(K_i,$ In$(K_i, O \cup \{o\})),$ In$(K_i, O \cup \{o\}))\}$
12 $\mathcal{U} \leftarrow$ Min($\mathcal{U}, \subseteq_{\mathcal{O}_i}$)
13 $\mathcal{L} \leftarrow \emptyset$
14 foreach $T \in MinTrans(MinGen(O))$ do
15     $\mathcal{L} \leftarrow \mathcal{L} \cup \{(O \setminus T,$ In$(K_i, O \setminus T))\}$
16 return $C, \mathcal{U}, \mathcal{R}, \mathcal{L}$
```

concept. We add the relational attributes to the previous intent. Our concept is now augmented with $\exists$ *Supports*.(C_DBMS_2) and $\exists$ *Supports*.(C_DBMS_3) in its intent.

The computation of the closed relational neighborhood comes after Line 6. We start with the relational cover. For each relational attribute in the updated intent, we add the corresponding concept in the relational cover. In our example, $\mathcal{R}$ takes C_DBMS_2 and C_DBMS_3.

The upper cover comes next. For each object that is not in the extent of our starting concept (here it is only MySQL Workbench), we compute the corresponding concept in the updated context. Our upper cover $\mathcal{U}$ receives the top concept, which intent has been updated: $\{(\top, \{$OS:Windows, $\exists$ *Supports*.(C_DBMS_3)$\})\}$ (concept C_DM_tools$^+$_10 from Figure 6.10).

For the lower cover, we want to find the minimal transversals of the set of minimal generator of O. The minimal generators of our extent are $\{$Astah, Erwin DM$\}$ and $\{$Astah, ER/Studio$\}$. This gives $\{$Astah$\}$ and $\{$Erwin DM, ER/Studio$\}$ as minimal transversals. When we compute the concept corresponding to O minus one minimal transversal of its minimal generators, we obtain the two following concepts that have respectively $\{$Erwin DM, ER/Studio, Magic Draw$\}$ and $\{$Astah, Magic Draw$\}$ as extent (concepts C_DM_tools$^+$_5 and C_DM_tools$^+$_6 of Figure 6.10).

The concept lattice that arises from the updated version of *DM_tools* is shown in Figure 6.10. Once a user has all the information available about the closed neighbor-

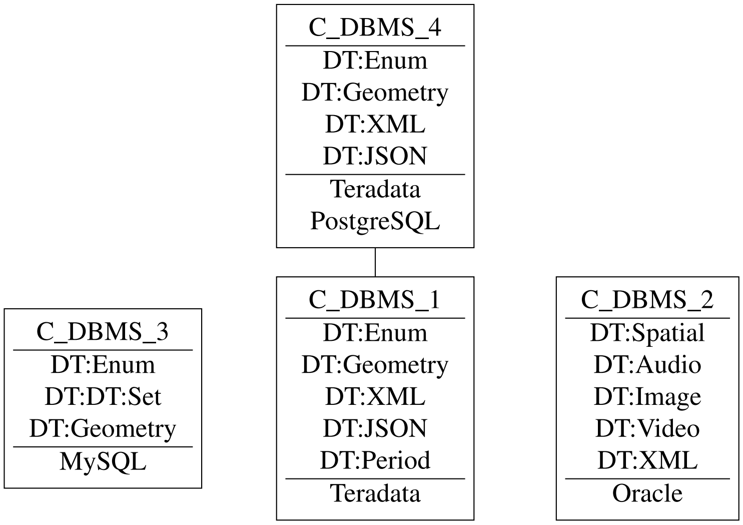


Figure 6.9: OC-poset of the context *DBMS*.

hood of their starting concept, they can either stop the exploration, or select a new starting concept in the closed neighborhood, from which a new step of exploration can start.

6.3.6 Discussion and future works

The algorithmic process from the previous section has been implemented in RCAExplore[5] by the Marel team at the LIRMM, with whom this was a joint work[6]. Some tests on real life datasets are in progress in order to measure the impact of the partial extension of the context and some time saving heuristics (that would allow to bypass the costly computation of the minimal transversals). It would be interesting to have a proof that the process converges in the same way that classical RCA does: are all concepts reachable in a finite number of steps from any starting concept?

[5] `http://dolques.free.fr/rcaexplore/`

[6] In particular with Jessie Carbonnel (for the partial generation of AOC-posets and on-demand RCA) and Marianne Huchard (for on-demand RCA). Alexandre Bazin was also part of the team on this work.

DM_tools$^+$	OS:Windows	OS:Mac OS	OS:Linux	DM:Conceptual	DM:Physical	DM:Logical	DM:ETL	$\exists$ sup.(C_DBMS_1)	$\exists$ sup.(C_DBMS_2)	$\exists$ sup.(C_DBMS_3)	$\exists$ sup.(C_DBMS_4)
Astah	×	×	×	×					×	×	
Erwin DM	×			×	×	×		×	×	×	×
ER/Studio	×			×	×	×	×	×	×	×	×
Magic Draw	×	×	×	×	×	×			×	×	×
MySQL Workbench	×	×	×		×					×	

Table 6.5: At the end of the first loop of our algorithm, the starting context has been extended with four new relational attributes and the relation has been updated.

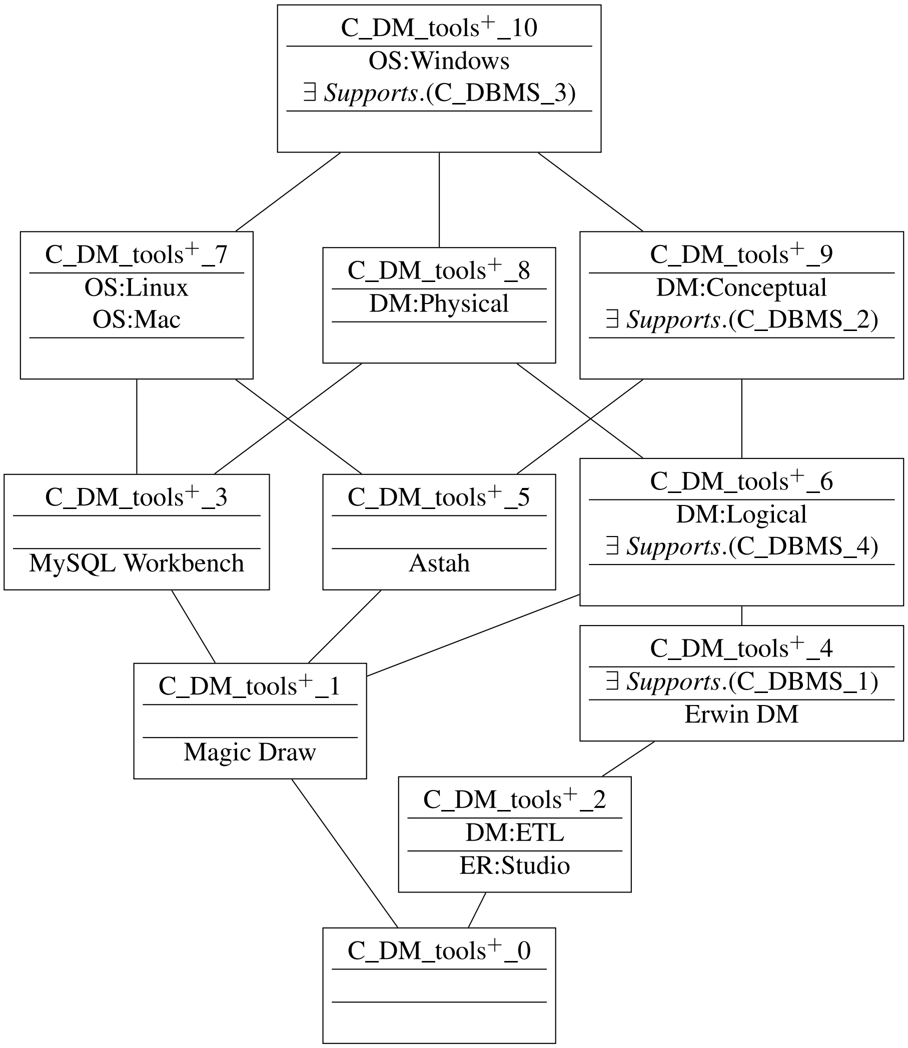


Figure 6.10: Concept lattice that corresponds to the augmented context *DM_tools*$^+$ (simplified representation).

Chapter 7

Closed-sets can tell who you are

This chapter, last of the more applied part of this dissertation deals with classification as a data mining task. It is organised as follows. First, we will describe why, as in which applicative context, we will use classification. Then, we present the formal framework in which we operate, which is closely related to both our applicative context and to the mathematics involved in all the previous chapters. Then, we introduce the model of our classifier and compare it to other models. Finally, we dive into an experimental study of classification of web browsing data.

7.1 Context

7.1.1 Identification and authentication, a heavy process

Virtually all the services that are provided to users have an online alternative. Think about banks, health, insurance companies, traval, commerce,etc. It is popularly acknowledged that online services lead to less time loss and stress that physical offices do. [1]

[1] Although online services have a lot of problems and limitations, they are not the object of this study.

Such an online interface with an agency poses problems of security. Banking or health data are sensitive data that we do not want anyone to have access to. This calls for enhanced security measures, in addition to the classical login and password couple. It is common for banks to have a secondary (physical) key that delivers either a precalculated key, or some proof of identity from the user. For example, when you are accessing an online banking system, the bank can send you a text message on your phone with an additional password. Such a process of authentication, that demands an action from the user, is called *explicit*.

This adds to the weight of a system: the compromise of security versus user-friendliness goes in the direction of security. This implies that more transactions are abandoned before reaching their end. In a case of a bank, this might not be the end of the world, but when the online task is inherent to the business model of a company, then this company cannot afford to let too many transactions go south.

On the other hand, when the system studies the user's behaviour in order to identificate him, or to validate some declared identity, then the authentication is said to be *implicit*: no actions from the user are needed to authenticate him, other than the actions he was already going to do on the system. Implicit authentication answers the question *"Am I who I claim to be ?"*, as stated in this paper about implicit authentication in mobile devices [64]. The answer to this question can be a clear yes or no, or a more temperate answer, with some degree of confidence. However the universe of possible answers is still binary.

A closely related problem is implicit identification. The question that we want to answer is now *"Who is this?"*. We do not have a binary output anymore, as we will see in section 7.3, where we have a pool of three hundred users to choose from.

We approach this problem by designing a classifier that guesses the corresponding user from a given anonymous behaviour. Through this classifier, we will be able to authenticate a user as follows: if the classifier correlates the right user behind the current anonymous behaviour, then the user is authenticated. Otherwise they may be rejected. Figure 7.1 shows the classical learning schema that is followed by our classifier.

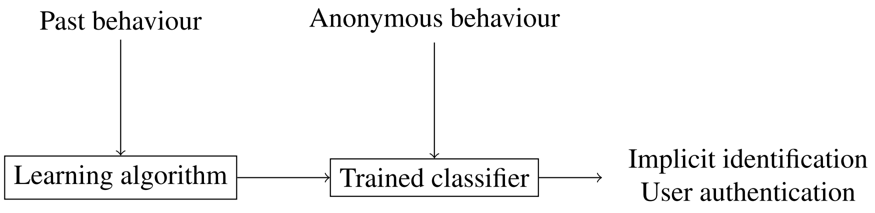


Figure 7.1: We start by training the classifier on some past behaviour. Then, from an anonymous behaviour, the trained classifier choses a class for this behaviour. If the class corresponds to the announced user, then we can authenticate them, if not, the user is rejected (or a stronger, more engaging means of authentication is asked).

7.1.2 Related works: implicit authentication and classification

Quickly after they became so widely used, mobilephones and smartphones picked up implicit authentication systems. In [65], the authors studied behaviour based on variables that are specific to smartphones (calls, texts, browsing between applications, location and time). Their experiments are based on the data for fifty users, over a period of twelve days. They gathered the data by using an application instaled by volonteers. The user profiles were built up from how frequently positive and negative events occurred, and their location. In this context, a positive event is an event consistent with the information gathered upsteam. As an example, calling a number which is in the phone's directory is a positive event. The results of this study show that based on around ten actions, one can detect fraudulent use of a smartphone with an accuracy of around 95%.

In a quite different context, in [66], the authors rely on Bayesian classification in order to associate a behaviour class with some video streaming user. Their simulated dataset consists of one thousand users over a course of one hundred days. They take into account the flow, the type of program, the duration of a session, the type of user and the popularity of the video. However, their results are not accurate enough for a security point of view, since their model has an accuracy of around fifty percent.

The particular context of implicit authentication in web browsing, which constitutes our case study, was studied in [67], [68], [69] and [70].

In [67], Yang designed a similar strategy and introduces a complete experimental protocol for authentication of web users. This study uses the domain name, the number of pages viewed, the session start time and its duration as characteristic variables. The dataset, which was gathered by a service provider, consists of the web-browsing history of 50,000 volonteers for over a year. The experiments are then carried on a subset of at most one hundred users. In her study, each class is described by a set of singletons with either the best support (number of appearances in the database) or the best lift (based on the number of appearance for a given class, compared to the number of appearances in the whole database). We included both of her heuristics in our experimental study. This study shows that for small anonymous behaviours that involve up to twenty websites, the most effective models are still the traditional classification models such as decision tree. On the other hand, whenever the anonymous behaviour exceeds seventy websites, the support-based and lift-based classifier are more efficient.

The study by Abramson and Aha [70], conducted on ten users over a one-month period, did not enable to build a significant model for distinguishing users. They conclude that no variable taken individually enables a user to be authenticated. Additionally, the size of the dataset remains a determining parameter: the learning phase needs time and a lot of data to be fed with.

In [71], drawing inspiration from Yang, Hermann, Banse and Federrath studied several techniques for analyzing a user who holds a dynamic IP address, based on behaviour models. They compare three methods: 1-nearest neighbour, multinominal Bayesian classifier and the same pattern mining technique based on support and lift used by Yang. They have records of DNS requests for three thousand and six hundred users over a two-month period. In this study, only the most significant variables and the most popular host names were considered. The accuracy rates for their models are satisfactory.

From our point of view, Yang's study [67] is very detailed, rigorous and accurate. For this reason, we faithfully reproduce here the author's experimental protocol. The strategy that we adopt is to identify for each user the list of patterns that best characterize their behaviour, but without limitation on the size (Yang allows only patterns of size one). We also handle closed sets in our classifier. Our objective is to show that it is more efficient to extract closed patterns of different size (essentially concept intents).

Our classifier is inspired by classification in concept lattices. More precisely, Kuznetsov designed, in [72] and [73], a classifier that works in concept lattices from positive and negative examples, by taking some inspiration in the work of Finn [74] that describe a plausible reasonning JSM type system. However, using directly this binary classifier in our context would label most of the anonymous behaviours as non-identifiable. For this reason, we adapted this classifier by using some more measures to select some particular patterns for each users, and the nearest neigbour method to take the final decision of the classification.

In 2007, Ramamohanarao and Han [75] defined *emerging patterns* as patterns that appear frequently on the objects of a single class, but are harder to find in other classes. For two surveys on emerging patterns, we advise the reader to consult [76] and [77]. A difficulty remains in selecting the most efficient emerging patterns among the large number of patterns. Ramamohanarao and Han [75] used the support to define which pattern are emerging. In our study, we show that the *tf-idf* parameter (mostly used in information retrieval, cf. [78]) is also efficient.

7.2 Model

In this section, we describe our formal model to perform implicit authentication. This model was introduced before my academic birth [2], and thus my contributions are more related to heuristics and experiments rather than to designing the model. Nevertheless, in order to have an understanding of the heuristics and why we even consider them, it is necessary to be familiar with the model as a whole.

7.2.1 Preliminaries and measures

We want to start from a list of events for each user. As we will see in the experiments section, each event in our data includes the user identifier, a time stamp and the name of the website that the user wanted to reach. First of all, we separate the data for each user to constitute a (multi)set of sessions [3]. The sessions are sets of websites that the user visited. We consider that the size of those sets is fixed [4]. Taking into account the time stamp associated to each entry, we can build the set of sessions associated to a

[2] Preliminary versions of this work started during Diyé Dia's thesis work at the LIMOS, and appeared in [79], [80] and [81].

[3] A user can have two or more identical sessions.

[4] In our experimental protocol, we fix the size of a session to be ten. This choice remains subjective and the size should be a natural parameter of the study.

Alex			Bill		
t_1	Alex	a	t'_1	Bill	d
t_2	Alex	b	t'_2	Bill	a
t_3	Alex	c	t'_3	Bill	g
t_4	Alex	d	t'_4	Bill	h
t_5	Alex	e	t'_5	Bill	a
t_6	Alex	c	t'_6	Bill	g
t_7	Alex	f	t'_7	Bill	d
t_8	Alex	b	t'_8	Bill	e
t_9	Alex	c	t'_9	Bill	a
t_{10}	Alex	d	t'_{10}	Bill	d
t_{11}	Alex	a	t'_{11}	Bill	e
t_{12}	Alex	e	t'_{12}	Bill	g

$$\begin{aligned}
\mathcal{S}_{\text{Alex}} &= \{\{a,b,c,d\},\{e,c,f,b\},\{c,d,a,e\}\} \\
\mathcal{S}_{\text{Bill}} &= \{\{d,a,g,h\},\{a,g,d,e\},\{a,d,e,g\}\} \\
X_{\text{Alex}} &= \{a,b,c,d,e,f\} \\
X_{\text{Bill}} &= \{a,d,e,g,h\} \\
\mathcal{X} &= \{a,b,c,d,e,f,g,h\}
\end{aligned}$$

Figure 7.2: Alex and Bill each have a list of events (represented by letters from the alphabet), that are labeled with some time stamps going from t_1 to t_{12} and t'_1 to t'_{12}. While building the sessions, we group the events by sets of four, keeping only the name of the website. Duplicates are deleted from sessions at this point (a succession of events $abacd$ becomes a four elements session $\{a,b,c,d\}$).

user according to the natural order of the events. However, we drop the ordering after this phase and consider the sessions to be sets.

For a user u, we denote by $\mathcal{S}_u$ the set of sessions associated to this particular user. We denote $\mathcal{S}$ the union of all the sessions of the database. Similarly, we call X_u the set of websites that are visited at least once by user u and $\mathcal{X}$ the set of all the websites of the database.

Let us consider a running example, that is presented in Figure 7.2. In our example, to ease the calculations, the sessions are of a fixed size four. We consider two users, Alex and Bill, that each have an associated list of events. These events are grouped in sets of four, depending on the time stamps. The ordering is, however, lost.

From now on, what we will call patterns are sets of websites that appear in a given session of a given user. We call a pattern a k-pattern when it has size k. In the following, we present the different measures that we will use to evaluate the quality of the patterns.

Definition 70 (Support and Lift)**.** *Let $\mathcal{S}$ be a set of sessions. The* support *of a pattern P in $\mathcal{S}$ is the percentage of sessions of $\mathcal{S}$ that contain P:*

$$Support_{\mathcal{S}}(P) = \frac{|\{S \in \mathcal{S} \mid P \subseteq S\}|}{|\mathcal{S}|}.$$

By extension, the support of a pattern P in the list of session $\mathcal{S}_u$ of a user u is

$$Support_u = \frac{|\{S \in \mathcal{S}_u \mid P \subseteq S\}|}{|\mathcal{S}_u|}.$$

For a given user u and a pattern P, the relative support *of P, for user u is*

$$Rel_u(P) = \frac{Support_u(P)}{Support_{\mathcal{S}}(P)}.$$

The support measures the strength of a pattern in behavioral description of a given user. The relative support mitigates the support by considering the pattern's support on the whole dataset. If a pattern appears a lot for every user, it is not characteristic and does not allow to discriminate between users.

For example, going back to our Figure 7.2, let us consider the patterns $\{a, d\}$ and $\{e, g\}$. The support of $\{a, d\}$ for Bill is high: $Support_{\text{Bill}}(\{a, d\}) = 1$, this pattern present in all three of Bill's sessions. The support of $\{e, g\}$ for Bill is lower: $Support_{\text{Bill}}(\{e, g\}) = 2/3$. However, when we mitigate these measures with the relative support, we can see that the pattern $\{e, g\}$, while having a smaller support in Bill's session, might be more characteristic of him. Let us compute the relative support for our two patterns, for Bill: $Rel_{\text{Bill}}(\{a, d\}) = 1/(5/6) = 1.2$ while $Rel_{\text{Bill}}(\{e, g\}) = (2/3)/(2/6) = 2$.

The *tf-idf* measure is a numerical statistic that is intended to reflect how relevant a word is to a document in a corpus of documents. The *tf-idf* value increases proportionally to the number of times a word appears in the document, but is counterbalanced by the frequency of the word in the whole corpus [78]. We adapt this measure by applying it to patterns, sessions and database.

Definition 71 (tf-idf)**.** *Let P be a pattern, U the set of all the users, and U_P the set of the users that have P in a least one of their sessions. For a given user u, the* normalized term frequency *(tf) of a pattern P, denoted $tf(P)$ is the support of P in this users' sessions. The* inverse document frequency *(idf) of a pattern P, denoted $idf(P)$ is* $\log(|U|/|U_P|)$*. Then the* tf-idf *of a pattern P, for a user u is*

$$tf \times idf_u(P) = Support_u(P) \times \log\left(\frac{|U|}{|U_P|}\right).$$

Going back to our running example. Since we have two users, $|U| = 2$. We consider the patterns $\{b, c\}$ and $\{a, d\}$. Since $\{b, c\}$ appears only in Alex's sessions, $|U_{\{b,c\}}| = 1$. For $\{a, d\}$, we have $|U_{\{a,d\}}| = 2$. Computing the *tf-idf* of both patterns for Alex gives us $tf \times idf_{\text{Alex}}(\{b, c\} = Support_{\text{Alex}}(\{b, c\}) \times \log 2 \approx 0.46$ and $tf \times idf_{\text{Alex}}(\{a, d\} = Support_{\text{Alex}}(\{a, d\}) \times \log 1 = 0$.

Until now, the patterns that we used to define our measures were simply sets. From now on, we want to handle closed patterns. We consider that a pattern is closed if it is an intent, that is if it cannot be extended with new websites while maintaining the same support (the size of the extent). We show the concept lattice associated with Alex's sessions in Figure 7.3.

Let P be a pattern. We denote by P^c the closure of P. In order to validate the use of the closed patterns, we will observe the following facts.

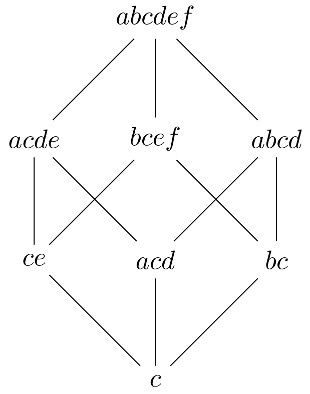


Figure 7.3: The concept lattice that corresponds to the lattice of closed patterns for Alex. Only the attributes are represented: since the objects are artificial session identifyiers, they do not carry useful information.

- Let P be a pattern, then $Support_u(P) = Support_u(P^c)$: by definition of our closure operator, the support does not decrease when applying the closure operator to a pattern in the set of sessions of a given user u.
- Let P be a pattern, then $Rel_u(P) \leq Rel_u(P^c)$.
- Let P be a pattern, then $tf \times idf_u(P) \leq tf \times idf_u(P^c)$.

It is noteworthy to mention that the support is an antimonotonic measure (that decreases when the pattern grows in size), while the relative support and the *tf-idf* are not.

7.2.2 Closed-set based classifier

The approach that we chose for our classifier is based on three points:

1. Pattern selection: from a dataset of web browsing logs, we compute a set of distinguishing patterns for each user.
2. Profile computation: from the patterns, we compute a vector, common to every user, and compute its component for each user. By this way, all the users are embedded in a common space.
3. Identification step: using the profile vectors, we search the nearest neigbour of an anonymous set of sessions using some similarity functions. We then compute confusion matrices and provide accuracy rates for our model.

Pattern selection

The pattern selection is the first and most important step in our model. It is this phase that changes depending on which heuristic we apply. We compute, for each user, their set of distinguishing patterns. This set is denoted $\mathcal{P}_u$ for user u.

The heuristics that we consider mainly change in their way of choosing the patterns for each user. In [67], each class is described by a set of 1-patterns with the best support or the best relative support. We call those two approaches H_{sup} and H_{Rel}. In our methods, we want to develop some heuristic using closed patterns. We consider three such heuristics: H^c_{tf-idf} that selects closed patterns based on their *tf-idf*, H^c_{sup} that considers closed patterns with the largest support and H^c_{supMin} that considers closed patterns that have the largest support and are minimal with respect to set inclusion (among the patterns for a user). We compared the performances of our heuristics and compiled the results in Figure 7.5. Those are fairly simple heuristics, that are based on simple measures of the "quality" of the patterns. In [82], Kuznetsov and Makhalova give a detailed survey on more complex measures of "quality" for concepts (such as stability, the amount of subset of the extent that are themselves extents, or robustness).

In the preliminary version of our tool, the closed patterns are computed using the Charm algorithm [83] provided by the Coron platform [84]. Since the tool implementing our classifier is more a proof of concept than a working tool, we do not use the monotony of the measures to prune the dataspace but rather compute all the closed patterns and choose the ones that correspond to our heuristics.

Profile computation

We define $\mathcal{P}_{all} = \bigcup_{u \in U} \mathcal{P}_u$ to be the set of all the patterns, for all the users. This set allows us to define a common space in which all the users may be embedded. In a more formal way, $\mathcal{P}_{all}$ defines a vector space $\mathcal{V}$ of size $|\mathcal{P}_{all}|$ where a given user u is represented by a vector $V_u = \langle m_{u,1}, \dots, m_{u,|\mathcal{P}_{all}|} \rangle$ where $m_{u,i}$ is a numerical value associated to the user u, for the pattern P_i, for i between 1 and $|\mathcal{P}_{all}|$ (Figure 7.4).

$$V_u = \begin{array}{|c|c|c|c|c|c|} \hline m_{u_1,1} & m_{u,2} & & \dots & & m_{u,|\mathcal{P}_{all}|} \\ \hline \multicolumn{1}{c}{P_1} & \multicolumn{1}{c}{P_2} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{\dots} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{P_{|\mathcal{P}_{all}|}} \end{array}$$

Figure 7.4: Each user is associated with such a vector, where numerical values are given for its descriptive patterns based on the support, the lift or the *tf-idf*.

That numerical value depends on the heuristic: we use the same measure here as the one that was used to choose the distinguishing patterns fo each user. Experiments confirm that the best accuracies are obtained when the measure that is used to choose the patterns and the one used to compute the profile concord.

Identification step

The identification step consists in trying to guess the user that corresponds to a set of anonymous sessions. All the sessions that are fed to the classifier in this step naturally come from the same user. For every set of anonymous sessions, we compute its numerical value for every component of the vector space.

The performances of our models are calculated on sets of anonymous sessions of growing size: from one session to thirty-five. Of course, the bigger our anonymous set is, the better the classification will be.

Let $\mathcal{S}_{anon}$ be a set of anonymous sessions. Using a distance function, we fill the vector $V_{anon} = \langle m_{anon,1}, \ldots, m_{anon,|\mathcal{P}_{all}|}\rangle$ with the measure that corresponds to the heuristic. We want to compute its distance to other profile vector.

In order to compute the distance between the profile vector, we present some similarity functions. Let $V_i = \langle m_{i,1}, \ldots, m_{i,|\mathcal{P}_{all}|}\rangle$ and $V_j = \langle m_{j,1}, \ldots, m_{j,|\mathcal{P}_{all}|}\rangle$ be two vectors. Let all be the size of $\mathcal{P}_{all}$.

The Euclidean distance between two vectors V_i and V_j, denoted $Euclidean(V_i,$ $V_j)$ is

$$Euclidean(V_i, V_j) = \sqrt{\sum_{\ell=1}^{all}(m_{j,\ell} - m_{i,\ell})^2}.$$

The Cosine similarity between two vectors V_i and V_j, denoted $Cosine(V_i, V_j)$ is

$$Cosine(V_i, V_j) = \frac{\sum_{\ell=1}^{all}(m_{j,\ell} - m_{i,\ell})}{\sqrt{\sum_{\ell=1}^{all}(m_{j,\ell})^2 \times \sum_{\ell=1}^{all}(m_{i,\ell})^2}}.$$

The Kulczynski similarity between two vectors V_i and V_j, denoted $Kulczyn$ $ski(V_i, V_j)$ is

$$Kulczinski(V_i, V_j) = \frac{\sum_{\ell=1}^{all} min(m_{i,\ell}, m_{j,\ell})}{\sum_{\ell=1}^{all} ||m_{j,\ell} - m_{i,\ell}||}, \ ||r|| \text{ denotes the absolute value of } r.$$

The Dice similarity between two vectors V_i and V_j, denoted $Dice(V_i, V_j)$ is

$$Dive(V_i, V_j) = \frac{2 \times \sum_{j=1}^{all}(m_{i,\ell} \times m_{j,\ell})}{\sum_{\ell=1}^{all}(m_{i,\ell})^2 + \sum_{\ell=1}^{all}(m_{j,\ell})^2}.$$

The performances of the heuristics depending on the similarity functions are compared in Figure 7.6 and 7.7.

7.2.3 Bayesian model

To serve as a “control group” for our experiments, we will compare all the heuristics to a Bayesian classifier. We recall that $\mathcal{S}$ is the set of all the sessions of the database, that can be mapped to our set of users. A Bayesian classifier states that an anonymous session S is assigned to a user i if and only if there is no user j different than i, such that $P(i \mid S) \geq P(j \mid S)$, with $P(A \mid B)$ the conditional probability that A, given that B. Bayes theorem tells us:

$$P(u \mid S) = \frac{P(S \mid u)P(u)}{P(S)}.$$

In practice, we are interested only in the term $P(S \mid u)$ of the fraction. For a session of ℓ websites s_1 to s_ℓ, under the assumption that websites are independent variables, $P(S \mid u)$ becomes $\prod_{i=0}^{\ell} P(s_i \mid u)$. For a given website s_i and user u, the probability $P(s_i \mid u)$ is calculated from the learning database. Since a website s_i is not a continuous value, we can define $P(s_i \mid u)$ as the number of sessions of user u that contain s_i, divided by the number of sessions of user u.

The major drawback with the traditionnal Bayes classifier is that a website that appears in an anonymous session but never occurs in the learning data will produce a probability equal to zero. To avoid this problem, we apply the Laplace smoothing (or add-one smoothing), which adds one to the support of each descriptor that appears in the dataset. This way, the unknown descriptor will have a support of one over the number of session, and all other descriptors will see their support augmented of the same amount. The frequency-based estimation of the probability will thus never be zero. From now on, when we refer to Bayes classifier, we are actually discussing the smoothed Bayes classifier.

7.3 Experiments

In this section, we give some experimental results. We start by describing precisely our dataset, which we consider as one of the contributions of this work. Then, we give a description of our experimental protocol and some results. We end on an analysis of the results and a discussion.

7.3.1 Description of our dataset

Our data comes from the proxy servers at the Blaise Pascal University. It was collected over a period of nine months during the school year 2013. It consists of seventeen million lines of connection logs from more than three thousand users and contains the user ID, a timestamp and a domain name for each entry. We applied two types of filters on the domain name in order to pre-process our data; our objective was to remove the entries that do not correspond to a deliberate action from the user. First, we applied blacklist filters. We used some lists of domain names considered as advertising [5], and removed the corresponding entries from our dataset. We also filtered the data using the status code obtained after a simple HTTP request on the domain name. [6] After those steps, our clean(ed) dataset has four million lines. Table 7.1 shows some key numbers about our dataset, before and after the preprocessing.

	#Users	#Sites	Avg #lines/user
Raw data	3388	96184	5082
After Preprocessing	3370	57654	1145

Table 7.1: Our pre-processing hopes to automatically remove actions that are not from the user.

We divided our file into the remaining three thousand users to obtain the class files. This dataset is available at `http://fc.isima.fr/~kahngi/cez13.`

[5] `http://winhelp2002.mvpe.org/hosts.htm` and `https://pgl.yoyo.org/as`

[6] This filter is more error prone, since we applied this filter one year and a half after the data was collected. Still, most of the domain names that were erased in this fashion were nonsensicals.

zip. The experimental study was conducted on the one hundred and fifty more active users.

7.3.2 Experiments

We fix a set of values for the number of anonymous sessions that we feed to our classifier. We test performances for the values one, three, five, ten, twenty, thirty and thirty-five. Clearly, the classification task is harder with less sessions.

Comparison between H^c_{tf-idf}, H^c_{sup} and H^c_{supMin}

We recall that H^c_{tf-idf}, H^c_{sup} and H^c_{supMin} are heuristics that consider only closed sets. The first uses the *tf-idf* as a discriminating measure, the second uses the support and the last uses the support and a test of inclusion. We executed those three heuristics on our dataset of 150 classes, using anonymous sets of sessions from one to thirty-five sessions. We fixed a limit on the number of patterns for each user (140 patterns) and their size (seven websites). Since the average size of a session is ten, patterns of size more than seven are rare. The results are smoothed over ten executions. Figure 7.5 compares the accuracy of the three heuristics based on closed-sets with the Bayes classifier.

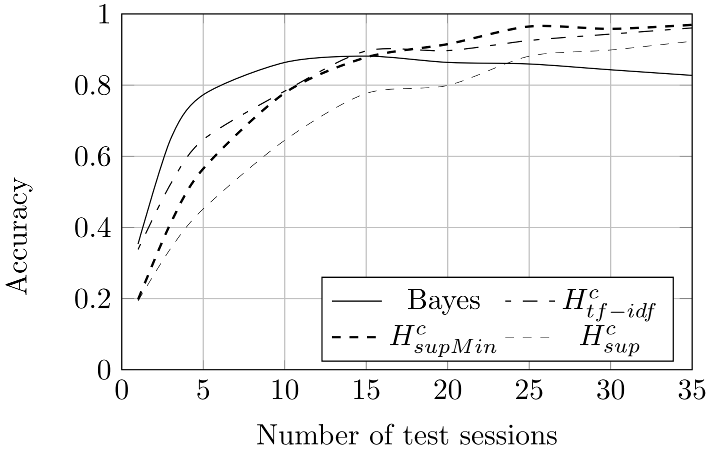


Figure 7.5: Comparative performance of H^c_{tf-idf}, H^c_{sup}, H^c_{supMin} and Bayes on the dataset of 150 users. The number of patterns per user is fixed to 140 and their maximal size to 7. Measured values are smoothed on 10 executions.

From Figure 7.5 we can see that the three heuristics based on closed-sets reach an accuracy of more than 90% when the size of the anonymous set of sessions is sufficiently large (beyond 25 sessions). We can also see that H^c_{tf-idf} and the Bayes classifier can succesfully classifier one user among 150 with only one session in around 30% of cases. This proportion reaches 50% for three sessions for H^c_{tf-idf}, and 60%

for Bayes. We can also see that H^c_{sup} has lower accuracy than the other heuristics, which back up the idea that support is not the best measure to be used on its own.

Comparison with H_{sup} and H_{Rel}

We compare our results with those by Yang [67], in Table 7.2. Table 7.2 shows results for three classes of methods: statistical (with the smoothed Bayesian methods), absolutes (H_{sup} and H^c_{sup} rely on the support, and absolute measure) and relative (H_{Rel} and H^c_{tf-idf}). We restrict our heuristics by only considering patterns of size one (for this reason, H^c_{supMin} is not represented). We can see that for 2, 10, 50, 100 or 150 classes, all the heuristics tend to have similar results with a high number of sessions. This validates that the behaviour of a user on the web, represented by the list of the websites they visited, allows to identify them. Since from around ten sessions, the accuracy of the classifier (whichever method and number of classes is considered) is beyond 80%, our challenge is then to identify a user from the smallest possible number of sessions.

When we only have one session, the Bayes classifier and H^c_{tf-idf} give similar accuracy rates, but are both outperformed by H_{Rel}. We can observe that the accuracy rates of the heuristics based on relative methods are significantly better than the results obtained with the support. Moreover, selecting the 1-patterns with best support is not less efficient than selecting the frequent closed-1-patterns, as shown by the comparability between H_{sup} and H^c_{sup}. If anything, H_{sup} is more efficient because it selects more patterns in this case.

Size matters

Table 7.3 shows the repartition of patterns by size, depending on the heuristic. We can see that H^c_{tf-idf} and H^c_{sup} prefer patterns of more than one element. For example, for heuristic H^c_{tf-idf}, the size of the pattern produces a normal distribution centered at three. This distribution is obtained when we allow 140 patterns per user.

Comparative performances when varying the distances

In Figure 7.6 and 7.7, we show the effect of the similarity measure on the accuracy rate. Figure 7.6 shows the results for heuristic H^c_{tf-idf} while Figure 7.7 shows the results for H^c_{supMin}.

Conclusion and discussion

In this chapter, we consider a case study of authentication of users from web-browsing logs. Our classifier and heuristics are based on closed-sets as descriptions of users. We compared ourselves to the results in the litterature, and obtained comparable results.

Since the challenge of implicit authentication finds its roots in a security concern, it is important to be able to recognize users from the least possible number of actions. In this regard, our classifier performs well, but has smaller accuracy rates than the statistical classifier Bayes, or the 1-pattern based classifier introduced by Yang (not

# Users	# Sessions	Bayes	H_{Sup}	H^c_{sup}	H_{Lift}	H^c_{tf-idf}
2	1	0.88	0.83	0.82	0.95(0.65)	**0.96** (0.63)
	10	0.97	**1**	**1**	0.95	**1**
	20	1	1	1	1	1
	30	1	1	1	1	1
10	1	0.71	0.55	0.53	**0.83**(0.64)	0.62
	10	**0.99**	0.88	0.88	0.94	0.97
	20	0.91	**1**	**1**	0.98	**1**
	30	0.96	**1**	**1**	**1**	**1**
50	1	0.47	0.31	0.30	**0.53**(0.85)	0.39
	10	0.90	0.89	0.88	0.87	**0.92**
	20	0.91	0.94	0.94	0.93	**0.99**
	30	0.88	0.98	0.98	0.98	**1**
100	1	0.40	0.25	0.24	**0.43**	0.33
	10	**0.88**	0.83	0.82	0.84	**0.88**
	20	0.88	0.95	0.95	0.92	**0.98**
	30	0.87	0.98	0.98	0.95	**1**
150	1	0.35	0.21	0.20	**0.39**	0.28
	10	**0.85**	0.80	0.78	0.79	**0.85**
	20	0.86	0.94	0.93	0.90	**0.98**
	30	0.84	0.98	0.97	0.92	**1**

Table 7.2: On the left we find the number of users and the size of the anonymous set of sessions. Sessions are of size 10. The accuracy rates are smoothed on 10 executions, the highest values are in bold. To compare fairly the different approaches we applied the best parameters to each one. This way, we used the *cosine* similarity for H_{Sup}, H^c_{Sup} and H_{Lift} and the *Kulczynski* similarity for H^c_{tf-idf}. All methods computes 1-patterns.

by a great margin, but still). However, Bayes classifier works as a black box, and Yang's 1-patterns lack expressivity. Closed patterns tend to be of a greater size than non-closed. This allows for a domain expert to be able to understand more of their system. In web-browsing logs, this is not a critical issue. However, that is not the only area where our work was considered. In a partnership with the company Almerys [7], the expressivity allowed by our classifier is a plus in a context of health insurance. As a part of the project that launched my thesis, there were discussions to apply this classifier and some learning methods, based on closed-sets, to some enterprise ressource planning softwares.

In this direction of expressivity of the data, one could ask why we used closed-sets in place of some particular closed-sets such as introducers concepts. In this case, we did not use introducer concepts because of the nature of the objects. In our context, the objects are artificial sessions, created arbitrarily to be of size ten. In order to use introducer concepts, objects need to have a semantic significance: in software engineering, objects are valid configuration that have a real role. To allow that, we

[7] http://www.be-almerys.com/

Heuristic	1	2	3	4	5	6	7
H^c_{tf-idf}	8.7%	25.8%	30.3%	20.8%	9.7%	3.5%	1.2%
H^c_{sup}	11.7%	33.6%	33.1%	15.5%	4.1%	0.8%	1.2%
H^c_{supMin}	87.6%	9.5%	1.8%	0.5%	0.16%	0.02%	0.01%

Table 7.3: For each heuristic, we give a repartition of the pattern by size. Recall that the size is user-defined to be bounded by seven. Those values are smoothed on 30 runs.

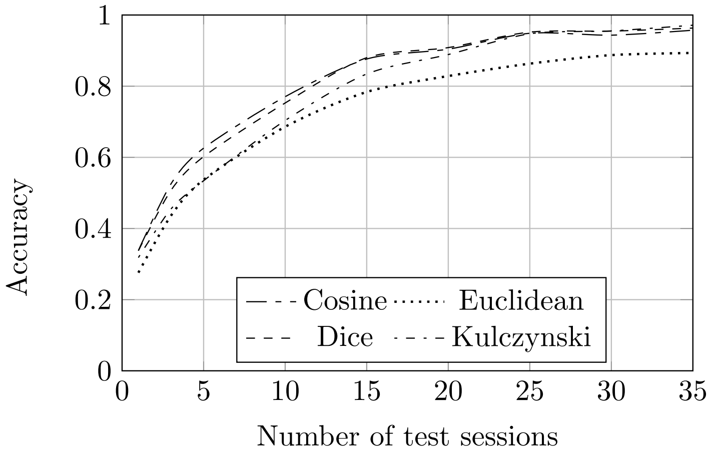


Figure 7.6: Comparison between the similarity measures, using H^c_{tf-idf}.

would need to use other ways of delimiting sessions, possibly with the time stamp, or with a physical identifier.

Speaking of arbitrary separation of the dataset, we can consider that separating users is a loss of information. We can consider that the whole dataset is in fact triadic: $users \times sessions \times websites$. With that in mind, and possibly a better partition of the logs into more meaningful sessions, we can for example extract 3-concepts in order to have some information that spans across users instead of separating them. The same model as before can still be kept using the introducer 3-concepts on the $users$ dimension.

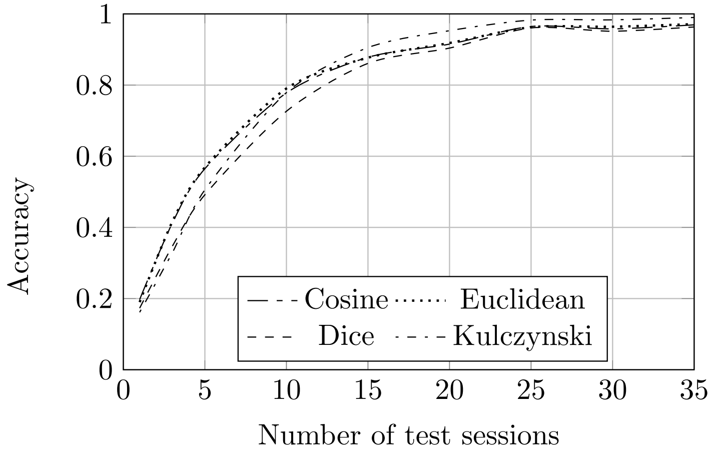


Figure 7.7: Comparison between the similarity measures, using H^{c}_{supMin}.

Conclusion

So, here we are, nearing the end. Our journey took us first in Combinatorial places, where we reflected upon the maximal size of d-lattices. We started from dimension three, to find a close interval where the actual value for the maximal size of a 3-lattice lives. The upper bound of this interval was computed using a method of measure and conquer, with an articulated proof based on a case study. The lower bound was constructed using a small 3-context with a lot of 3-concepts, and by multiplying its the 3-concepts with a product of contexts. This construction allowed us to consider another view of powerset d-lattices.

This work has left some unanswered questions. What is the actual maximal size for a 3-lattice (we conjectured that it might be $\left(\frac{27}{8}\right)^n$ for an $n \times n \times n$ context)? Is there a more automated way of computing bounds for dimension d?

Our next stop, still in Combinatorial waters, was the average case. We discussed the average number of implications in some implication bases and the number of concepts in two dimensions, and later in d-dimensions. We also discussed the pitfalls of randomly generating contexts, a subject that is of increasing concern in the FCA community.

The main question that remains open is the average size of the Duquenne-Guigues implication base, the smallest base cardinality-wise. The approach that we used cannot be used to compute the average size of the Duquenne-Guigues base. Indeed, the recursive nature of pseudo-intents makes approximating their number on average a difficult task.

We then entered the Algorithms part of the journey. Once again, this part was divided into two chapters. In the first chapter, we introduced an incremental algorithm to compute the d-concepts of a d-context. This algorithm considers the not-so-usual task of computing the d-concepts of a d-context after a new element has been added to the d-context. Once this problem was solved, there was a natural algorithm to compute all the d-concepts of a d-concepts from scratch.

The possible extensions of this work would be a comparison between our algorithm and the other algorithms from the literature. Moreover, either by changing our algorithm or by using a totally novel approach, it would be interesting to compute the order structure that comes with the elements of the d-lattice.

The second chapter of the Algorithms realm did not propose an algorithm strictly speaking. Instead, we studied reduction and introducer d-concepts. Those concepts play a key role in many applications in two dimensions, mainly by reducing the complexity and the number of objects to consider.

The obvious continuation of this work is to try to find applications in which we could use introducer concepts. Usually, a backward step allows to climb in abstraction, as we have seen in the previous chapter, where all the datasets induced by the users can be seen as a unique three dimensional dataset. Since two dimensional introducer concepts are mainly used in software engineering, it might be the best place to start to find meaningful applications of multidimensional introducer concepts.

When we entered the last part of the dissertation, called Applications, we vowed not to leave data and applied uses of our lattices behind. The chapter part concerned exploration. Data exploration allows to browse a potentially enormous universe without actually having to generate it fully. We started by looking more closely at the poset of introducer 2-concepts. In a context of software engineering, we developed an algorithm that retrieves the upper cover and the lower cover of any set of *features* (basically attributes). We also extended this exploration strategy to relational lattice families, and developed some algorithm to browse a family of lattices induced by RCA.

This work still has some open questions. Does our version of RCA converge to a fixed point? Is it really applicable to real life datasets? We hope to be able to answer them in a close future.

Our last chapter dealt with classification of users from web-browsing logs. We used 2-concepts to build profiles for users and used those profiles to recognize anonymous behaviours. Our results are comparable in terms of performances with the literature.

The next step in this matter is to try to implement a classifier using 3-concepts, in order to group users by their behaviour. We would also like to develop a cartography of users depending of their behaviour in a system. This approach is of obvious interest for many real life applications and businesses.

In the course of this dissertation, we saw that lattices and their multidimensional generalization are objects of interest both theoretically and in data analysis. Their central position in several problems of data analysis puts them into focus for most of theoretical studies in data sciences. Plus, lattices are cool.

Bibliography

[1] Garrett Birkhoff. *Lattice theory*, volume 25. American Mathematical Soc., 1940.

[2] Marc Barbut and Bernard Monjardet. Ordre et classification. *Algebre et Combinatoire*, 2, 1970.

[3] Bernhard Ganter and Rudolf Wille. *Formal concept analysis - mathematical foundations*. Springer, 1999.

[4] Fritz Lehmann and Rudolf Wille. A triadic approach to formal concept analysis. In *Conceptual Structures: Applications, Implementation and Theory, Third International Conference on Conceptual Structures, ICCS '95, Santa Cruz, California, USA, August 14-18, 1995, Proceedings*, pages 32–43, 1995.

[5] Rudolf Wille. The basic theorem of triadic concept analysis. *Order*, 12(2):149–158, 1995.

[6] George Voutsadakis. Polyadic concept analysis. *Order*, 19(3):295–304, 2002.

[7] David H. Krantz, Patrick Suppes, and Robert Duncan Luce. *Foundations of measurement*. academic press, 1971.

[8] Utta Wille. *Geometric representation of ordinal contexts*. PhD thesis, University Gießen, 1995.

[9] Norman L Biggs. The roots of combinatorics. *Historia Mathematica*, 6(2):109–136, 1979.

[10] Claude Berge and Edward Minieka. Graphs and hypergraphs. 1973.

[11] Thomas Eiter and Georg Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304, 1995.

[12] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, and Hannu Toivonen. Data mining, hypergraph transversals, and machine learning. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona, USA*, pages 209–216, 1997.

[13] Oliver Kullmann. New methods for 3-sat decision and worst-case analysis. *Theor. Comput. Sci.*, 223(1-2):1–72, 1999.

[14] Zbigniew Lonc and Mirosław Truszczyński. On the number of minimal transversals in 3-uniform hypergraphs. *Discrete Mathematics*, 308(16):3668–3687, 2008.

[15] Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A measure & conquer approach for the analysis of exact algorithms. *J. ACM*, 56(5):25:1–25:32, 2009.

[16] Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer Science & Business Media, 2010.

[17] Sergei O. Kuznetsov. On the intractability of computing the Duquenne-Guigues base. *J. UCS*, 10(8):927–933, 2004.

[18] William Ward Armstrong. Dependency structures of database relationship. *Information processing*, pages 580–583, 1974.

[19] Jean-Louis Guigues and Vincent Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences humaines*, 95:5–18, 1986.

[20] Sergei O. Kuznetsov and Sergei A. Obiedkov. Some decision and counting problems of the Duquenne-Guigues basis of implications. *Discrete Applied Mathematics*, 156(11):1994–2003, 2008.

[21] Mikhail A. Babin and Sergei O. Kuznetsov. Recognizing pseudo-intents is conp-complete. In *Proceedings of the 7th International Conference on Concept Lattices and Their Applications, Sevilla, Spain, October 19-21, 2010*, pages 294–301, 2010.

[22] Felix Distel and Baris Şertkaya. On the complexity of enumerating pseudo-intents. *Discrete Applied Mathematics*, 159(6):450–466, 2011.

[23] Mikhail A. Babin and Sergei O. Kuznetsov. Computing premises of a minimal cover of functional dependencies is intractable. *Discrete Applied Mathematics*, 161(6):742–749, 2013.

[24] Karell Bertet and Bernard Monjardet. The multiple facets of the canonical direct unit implicational basis. *Theor. Comput. Sci.*, 411(22-24):2155–2166, 2010.

[25] Uwe Ryssel, Felix Distel, and Daniel Borchmann. Fast algorithms for implication bases and attribute exploration using proper premises. *Ann. Math. Artif. Intell.*, 70(1-2):25–53, 2014.

[26] Michael L. Fredman and Leonid Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *J. Algorithms*, 21(3):618–628, 1996.

[27] Julien David, Loïck Lhote, Arnaud Mary, and François Rioult. An average study of hypergraphs and their minimal transversals. *Theor. Comput. Sci.*, 596:124–141, 2015.

[28] Paul Erdős and Alfréd Rényi. On Random Graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[29] Bernhard Ganter and Sergei A. Obiedkov. Implications in triadic formal contexts. In *Conceptual Structures at Work: 12th International Conference on Conceptual Structures, ICCS 2004, Huntsville, AL, USA, July 19-23, 2004. Proceedings*, pages 186–195, 2004.

[30] Alexandre Bazin. On implication bases in n-lattices. 2017.

[31] Sergei O. Kuznetsov. On computing the size of a lattice and related decision problems. *Order*, 18(4):313–321, 2001.

[32] Loïck Lhote, François Rioult, and Arnaud Soulet. Average number of frequent and closed patterns in random databases. In *Actes de CAP 05, Conférence francophone sur l'apprentissage automatique - 2005, Nice, France, du 31 mai au 3 juin 2005*, pages 345–360, 2005.

[33] Loïck Lhote, François Rioult, and Arnaud Soulet. Average number of frequent (closed) patterns in Bernouilli and Markovian databases. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 713–716, 2005.

[34] Richard Emilion and Gérard Lévy. Size of random galois lattices and number of closed frequent itemsets. *Discrete Applied Mathematics*, 157(13):2945–2957, 2009.

[35] Mikhail Klimushkin, Sergei A. Obiedkov, and Camille Roth. Approaches to the selection of relevant concepts in the case of noisy data. In *Formal Concept Analysis, 8th International Conference, ICFCA 2010, Agadir, Morocco, March 15-18, 2010. Proceedings*, pages 255–266, 2010.

[36] Alexandre Bazin, Olivier Bodini, and Julien David. Invalidating a conjecture on the average number of closed sets in a random database. 2018.

[37] Sergei O. Kuznetsov and Sergei A. Obiedkov. Comparing performance of algorithms for generating concept lattices. *J. Exp. Theor. Artif. Intell.*, 14(2-3):189–216, 2002.

[38] Daniel Borchmann and Tom Hanika. Some experimental results on randomly generating formal contexts. In *Proceedings of the Thirteenth International Conference on Concept Lattices and Their Applications, Moscow, Russia, July 18-22, 2016.*, pages 57–69, 2016.

[39] Bernhard Ganter. Random extents and random closure systems. In *Proceedings of The Eighth International Conference on Concept Lattices and Their Applications, Nancy, France, October 17-20, 2011*, pages 309–318, 2011.

[40] Andrei Rimsa, Mark A. J. Song, and Luis E. Zárate. Scgaz - A synthetic formal context generator with density control for test and evaluation of FCA algorithms. In *IEEE International Conference on Systems, Man, and Cybernetics, Manchester, SMC 2013, United Kingdom, October 13-16, 2013*, pages 3464–3470, 2013.

[41] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Closed patterns meet *n*-ary relations. *TKDD*, 3(1):3:1–3:36, 2009.

[42] Tatiana Makhalova and Lhouari Nourine. An incremental algorithm for computing n-concepts. In *Formal Concept Analysis for Knowledge Discovery. Proceedings of International Workshop on Formal Concept Analysis for Knowledge Discovery (FCA4KD 2017), Moscow, Russia*, 2017.

[43] Sebastian Rudolph, Christian Sacarea, and Diana Troanca. Reduction in triadic data sets. In *Proceedings of the 4th International Workshop "What can FCA do for Artificial Intelligence?", FCA4AI 2015, co-located with the International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 25, 2015.*, pages 55–62, 2015.

[44] Robert Godin and Hafedh Mili. Building and Maintaining Analysis-Level Class Hierarchies Using Galois Lattices. In *8th Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pages 394–410, 1993.

[45] Klaus Biedermann. Powerset trilattices. In *Conceptual Structures: Theory, Tools and Applications, 6th International Conference on Conceptual Structures, ICCS '98, Montpellier, France, August 10-12, 1998, Proceedings*, pages 209–224, 1998.

[46] Klaus Pohl, Günter Böckle, and Frank van der Linden. *Software Product Line Engineering - Foundations, Principles, and Techniques*. Springer, 2005.

[47] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Citeseer, 1990.

[48] Robert Godin, Jan Gecsei, and Claude Pichet. Design of a Browsing Interface for Information Retrieval. In *12th International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 32–39, 1989.

[49] Claudio Carpineto and Giovanni Romano. Exploiting the Potential of Concept Lattices for Information Retrieval with CREDO. *Journal of Universal Computer Science*, 10(8):985–1013, 2004.

[50] Jon Ducrou, Björn Vormbrock, and Peter W. Eklund. FCA-Based Browsing and Searching of a Collection of Images. In *14th International Conference on Conceptual Structures (ICCS), Conceptual Structures: Inspiration and Application*, pages 203–214, 2006.

[51] Mehwish Alam, Thi Nhu Nguyen Le, and Amedeo Napoli. LatViz: A New Practical Tool for Performing Interactive Exploration over Concept Lattices. In *13th International Conference on Concept Lattices and Their Applications (CLA)*, pages 9–20, 2016.

[52] Jon Ducrou and Peter W. Eklund. SearchSleuth: The Conceptual Neighbourhood of an Web Query. In *5th International Conference on Concept Lattices and Their Applications (CLA)*, 2007.

[53] Cássio A. Melo, Bénédicte Le Grand, and Marie-Aude Aufaure. Browsing Large Concept Lattices through Tree Extraction and Reduction Methods. *International Journal of Intelligent Information Technologies (IJIIT)*, 9(4):16–34, 2013.

[54] Simon Andrews and Laurence Hirsch. A tool for creating and visualising formal concept trees. In *5th Conceptual Structures Tools & Interoperability Workshop (CSTIW 2016) held at the 22nd Int. Conf. on Conceptual Structures (ICCS 2016)*, pages 1–9, 2016.

[55] Stéphanie Chollet, Vincent Lestideau, Yoann Maurel, Etienne Gandrille, Philippe Lalanda, and Olivier Raynaud. Practical use of formal concept analysis in service-oriented computing. In *Formal Concept Analysis - 10th International Conference, ICFCA 2012, Leuven, Belgium, May 7-10, 2012. Proceedings*, pages 61–76, 2012.

[56] Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing iceberg concept lattices with Titanic. *Data Knowledge Engineering (DKE)*, 42(2):189–222, 2002.

[57] Gillian J. Greene and Bernd Fischer. Single-Focus Broadening Navigation in Concept Lattices. In *3rd Workshop on Concept Discovery in Unstructured Data, co-located with the 13th International Conference on Concept Lattices and Their Applications (CLA)*, pages 32–43, 2016.

[58] Raoul Medina, Lhouari Nourine, and Olivier Raynaud. Interactive association rules discovery. In *Formal Concept Analysis, 4th International Conference, ICFCA 2006, Dresden, Germany, February 13-17, 2006, Proceedings*, pages 177–190, 2006.

[59] Anne Berry, Marianne Huchard, Ross M. McConnell, Alain Sigayret, and Jeremy P. Spinrad. Efficiently Computing a Linear Extension of the Subhierarchy of a Concept Lattice. In *3rd Int. Conf. on Formal Concept Analysis (ICFCA)*, pages 208–222, 2005.

[60] Mohamed Rouane Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. A Proposal for Combining Formal Concept Analysis and Description Logics for Mining Relational Data. In *Proc. of the 5th Int. Conf. on Formal Concept Analysis (ICFCA'07)*, pages 51–65, 2007.

[61] Mohamed Rouane Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. Soundness and completeness of relational concept analysis. In *Formal Concept Analysis, 11th International Conference, ICFCA 2013, Dresden, Germany, May 21-24, 2013. Proceedings*, pages 228–243, 2013.

[62] Mohamed Rouane Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. Relational concept analysis: mining concept lattices from multirelational data. *Ann. Math. Artif. Intell.*, 67(1):81–108, 2013.

[63] Xavier Dolques, Florence Le Ber, and Marianne Huchard. Aoc-posets: a scalable alternative to concept lattices for relational concept analysis. In *Proceedings of the Tenth International Conference on Concept Lattices and Their Applications, La Rochelle, France, October 15-18, 2013.*, pages 129–140, 2013.

[64] Tobias Stockinger. Implicit authentication on mobile devices. Technical report, 2011.

[65] Elaine Shi, Yuan Niu, Markus Jakobsson, and Richard Chow. Implicit authentication through learning user behavior. In *Information Security - 13th International Conference, ISC 2010, Boca Raton, FL, USA, October 25-28, 2010, Revised Selected Papers*, pages 99–113, 2010.

[66] Ihsan Ullah, Guillaume Doyen, Grégory Bonnet, and Dominique Gaïti. A bayesian approach for user aware peer-to-peer video streaming systems. *Sig. Proc.: Image Comm.*, 27(5):438–456, 2012.

[67] Yinghui (Catherine) Yang. Web user behavioral profiling for user identification. *Decision Support Systems*, 49(3):261–271, 2010.

[68] Ravi Kumar and Andrew Tomkins. A characterization of online browsing behavior. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 561–570, 2010.

[69] Sharad Goel, Jake M. Hofman, and M. Irmak Sirer. Who does what on the web: A large-scale study of browsing behavior. In *Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*, 2012.

[70] Myriam Abramson and David W. Aha. User authentication from web browsing behavior. In *Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2013, St. Pete Beach, Florida, USA, May 22-24, 2013.*, 2013.

[71] Dominik Herrmann, Christian Banse, and Hannes Federrath. Behavior-based tracking: Exploiting characteristic patterns in DNS traffic. *Computers & Security*, 39:17–33, 2013.

[72] Sergei O. Kuznetsov. Machine learning and formal concept analysis. In *Concept Lattices, Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004, Proceedings*, pages 287–312, 2004.

[73] Sergei O. Kuznetsov. Complexity of learning in concept lattices from positive and negative examples. *Discrete Applied Mathematics*, 142(1-3):111–125, 2004.

[74] Victor K. Finn. Plausible reasoning in systems of jsm type. *Itogi Nauki i Tekhniki, Seriya Informatika*, 15:54–101, 1991.

[75] Kotagiri Ramamohanarao and Hongjian Fan. Patterns based classifiers. *World Wide Web*, (1):71–83, 2007.

[76] Petra Kralj Novak, Nada Lavrac, and Geoffrey I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403, 2009.

[77] Milton García-Borroto, José Francisco Martínez Trinidad, and Jesús Ariel Carrasco-Ochoa. A survey of emerging patterns for supervised classification. *Artif. Intell. Rev.*, 42(4):705–721, 2014.

[78] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[79] Diyé Dia, Olivier Coupelon, Yannick Loiseau, and Olivier Raynaud. Suis-je celui que je prétends être ? In *IC 2014 : 25es Journées francophones d'Ingénierie des Connaissances (Proceedings of the 25th French Knowledge Engineering Conference), Clermont Ferrand, France, May 12-16, 2014.*, pages 271–274, 2014.

[80] Olivier Coupelon, Diyé Dia, Fabien Labernia, Yannick Loiseau, and Olivier Raynaud. Using closed itemsets for implicit user authentication in web browsing. In *Proceedings of the Eleventh International Conference on Concept Lattices and Their Applications, Košice, Slovakia, October 7-10, 2014.*, pages 131–144, 2014.

[81] Olivier Coupelon, Diyé Dia, Yannick Loiseau, and Olivier Raynaud. Am I really who I claim to be? In *New Contributions in Information Systems and Technologies - Volume 1 [WorldCIST'15, Azores, Portugal, April 1-3, 2015].*, pages 613–616, 2015.

[82] Sergei O. Kuznetsov and Tatiana P. Makhalova. On interestingness measures of formal concepts. *Inf. Sci.*, 442-443:202–219, 2018.

[83] Mohammed Javeed Zaki and Ching-Jui Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Trans. Knowl. Data Eng.*, 17(4):462–478, 2005.

[84] Laszlo Szathmary. *Symbolic Data Mining Methods with the Coron Platform. (Méthodes symboliques de fouille de données avec la plate-forme Coron)*. PhD thesis, Henri Poincaré University, Nancy, France, 2006.