Cadre général pour la recherche d'information et l'extraction de connaissances par l'exploration de treillis

Jean-François Viaud

Université de La Rochelle École doctorale S2IM

Cadre général pour la recherche d'information et l'extraction de connaissances par l'exploration de treillis.

Thèse

présentée et soutenue publiquement le 31 Août 2017
pour l'obtention du

Doctorat de l'Université de La Rochelle
(mention informatique)

par

Jean-François Viaud

Composition du jury

Président	Amedeo Napoli	Directeur de recherche CNRS, LORIA
Rapporteurs	Sergei O. Kuznetsov	Professeur, Higher School of Economics (Moscou)
	Lhouari Nourine	Professeur, Université Blaise Pascal Clermont-Ferrand
Examinateurs	Nathalie Caspard	Maitre de conférences, ESIAG
	Léonard Kwuida	Professeur, Berner Fachhochschule (Suisse)
	Jean-Loup Guillaume	Professeur, Université de La Rochelle, L3I
Directrice	Karell Bertet	Maître de conférences (HDR), Université de La Rochelle, L3I

Laboratoire Informatique, Image, Interaction (L3i) – EA 2118

A ma grand-mère. Tu es la première, pour ne pas être obligée de lire la suite.
A ma maman. Quand nous étions petits, Agnès et moi t'appelions "Madame Range-Tout" et il est surtout question de rangement.
A mon père. S'il y en a une troisième, elle sera de médecine.
A Ingrid, Agathe et Eléanor qui me supportent au quotidien. Vous seules savez à quel point cela peut être difficile.

Remerciements

Je commence par remercier tous les oubliés, qui ne se trouvent pas dans la liste ci-dessous. Ainsi je suis certain de n'avoir oublié personne.

Le premier nom sera Vincent C., car c'est un peu par toi que tout a commencé. Lorsque tu m'as proposé de faire partie de ton équipe, tu ne te doutais pas que ça m'emmènerait ici, avec en chemin un bouquin. Honnêtement, moi non plus. Merci Bertrand V., tu as été mon premier maître et non des moindres. Mon initiation aux réseaux, à la compilation, à tes côtés, a été un réel plaisir. Merci Mike E. de m'avoir mis sur la voie des bases de données, et aussi pour tout le reste, bien avant. Bien sûr, je remercie toute l'équipe d'Informatique Transversale, pour l'ambiance chaleureuse dans laquelle nous travaillons.

Je remercie Jean-Marc O., pour m'avoir chaleureusement accueilli au sein de son laboratoire, alors que je n'étais qu'un enseignant de mathématiques. J'espère renforcer ce lien entre les deux communautés que tu as permis de créer. Je remercie également tous les membres du L3I; vous être trop nombreux pour vous citer tous, mais le coeur y est. Je trouve l'ambiance dans le laboratoire propice à l'épanouissement.

Merci à toi, Laurent L.F., les échanges que nous avons régulièrement dans notre bureau sont toujours enrichissants. Merci Laurence C. pour le soutien très fréquent que tu apportes à tous les membres du département de mathématiques, moi inclus. Je remercie également tous les membres du département de mathématiques avec qui je travaille depuis environ dix ans.

Je remercie la communauté FCA présente à ICFCA2015 à Malaga. L'accueil chaleureux m'a permis de me sentir très vite intégré et tous les échanges que nous avons eus sont une immense richesse.

Merci Karell B. d'avoir accepté d'encadrer cette thèse, de m'avoir guidé tout au long de son élaboration. Ton accompagnement non seulement scientifique mais aussi et surtout humain a été réellement précieux. Merci Rokia M. pour tout, mais en particulier pour le séjour à l'UQO; ton accueil a vraiment été fantastique. Enfin, merci Christophe D., là encore, pas seulement pour les aspects techniques autour de notre projet, mais aussi pour tous les conseils et les échanges que je souhaite continuer à avoir.

Enfin, j'embrasse ma famille toujours proche, même lorsqu'elle est loin. Encore merci à Ingrid, Agathe et Eléanor qui partagent mon quotidien.

Abstract

During the last two decades, data have literrally overwhelmed the world. Indeed a huge amount of heterogenous data is daily produced, so that techniques of Information Retrieval have to evolve to order them and select relevant ones.

On the other side, techniques of Knowledge Discovery are able to extract a potentially exponential number of patterns from data, especially association rules, so that new tools have to be defined to help data analysts in their job.

Both information retrieval and knowledge discovery address the same issue : they structure and organize data. Nevertheless their points of view are different: the former selects and ranks data whether the latter classifies and clusters them.

Formal Concept Analysis (FCA), introduced by R. Wille, uses concept lattices to reveal both an order and a classification inside data. However, it is well known in the FCA community, that these concept lattices may have an exponential size with respect to data.

For all these reasons, tools to reduce the size of data, or lattices, are needed. In this thesis, some distributed algorithms for FCA have been designed in order to reduce input data into small pieces. Different decompositions of lattices have also been studied or defined, some based on congruence relations, other on tolerance relations. At last, to help the user in his choices of reduction, a general framework, named LattExp, have been defined.

LattExp provides a navigation facility through reductions/decompositions and guide the user in his choices.

Résumé

Au cours des deux dernières décennies, nous avons été littéralement submergés par les données. Nous recevons, rapidement, un flux colossal de données hétérogènes. Les techniques issues de la recherche d'information doivent évoluer afin de les ordonner et de les sélectionner.

D'un autre côté, les techniques d'extraction de connaissances sont capables d'extraire une quantité (exponentielle) de connaissances des données, en particulier lors de la fouille de règles d'association. De nouveaux outils doivent être fournis aux analystes des données.

La recherche d'information et l'extraction de connaissances abordent la même problématique : elles structurent et organisent les données. La première sélectionne et attribut un rang, l'autre partitionne et hiérarchise.

L'Analyse des Concepts Formels (FCA), introduite par R. Wille, utilise les treillis de concepts afin de mettre en évidence à la fois un ordre (au sens d'attribuer un rang) et une classification (au sens de partition). Cependant, il est bien connu de la communauté FCA que ces treillis peuvent avoir une taille exponentielle.

Pour toutes ces raisons, des outils pour réduire la taille des données ou des treillis, sont nécessaires. Dans cette thèse, nous avons utilisé différentes approches. Des algorithmes distribués pour la FCA ont été définis afin de réduire la taille des données d'entrée en petits morceaux. Différentes méthodes de décomposition de treillis ont été étudiées ou définies. Certaines sont basées sur les relations de congruence, d'autres sur les relations de tolérance. Pour aider l'utilisateur dans ses choix de réduction, un cadre général, nommé LattExp, a été mis en place. LattExp fournit une possibilité de navigation à travers les réductions/décompositions et aide l'utilisateur dans ses choix.

Table des matières

Première partie

Présentation

Chapitre 1

Introduction

Sommaire

1.1 Big input, big output

Au cours des deux dernières décennies, nous avons été littéralement submergés par les données. Nous sommes progressivement entrés dans l'ère des objets connectés et ces objets transmettent en continu des données. Nous recevons ainsi, rapidement, un flux colossal de données hétérogènes. C'est le "Big Input" auquel doivent faire face les techniques de Recherche d'Information (*Information Retrieval* IR), afin d'ordonner et de sélectionner ces données.

D'un autre côté, les techniques d'Extraction de Connaissance (EC) sont capables d'extraire une quantité exponentielle de connaissances [1] des données, en particulier lors de l'extraction de règles d'association. C'est le "Big Output" auquel doivent faire face les analystes des données dans leur travail.

De ce "Big Input" à ce "Big Output", nombreux sont les nouveaux challenges à relever. L'accent est mis sur la sémantique des données, par exemple dans le cadre du traitement du langage naturel ou pour le web sémantique [6]. Les méthodes de réduction de dimension évoluent, de nouvelles visualisations sont

[1] A priori, ces connaissances sont extraites pour l'humain ; a posteriori, pour faire face à la quantité de connaissances extraites, la machine doit retraiter ces connaissances pour être exploitées par l'humain.

produites et de nouveaux champs de recherche sont explorés, comme l'apprentissage profond.

Cette thèse aborde, à sa toute petite échelle[2], les deux problèmes du point de vue de l'analyse des concepts formels.

1.2 Grands jeux de données

Le Big Data (données massives) est en général caractérisé, ou défini, par les trois "V" : il s'agit de données Volumineuses dont la Vitesse d'évolution est élevée et qui sont d'une très grande Variété. Le Volume ne sera pas détaillé, même s'il existe une échelle dans la complexité liée à la taille des données. Disons qu'un projet qui travaille en exabyte peut revendiquer être dans le cadre des données massives. La vélocité signifie d'une part, par exemple, que les objets connectés produisent des données en continu, il faut donc les traiter à la volée, immédiatement. Vélocité fait également référence à l'évolution : les données changent rapidement au cours du temps. Enfin, la variété désigne la présence de structures multiples. Il s'agit, par exemple, de données produites par des sources différentes.

Les données massives sont un enjeu de société majeur d'ici 2020 [28, 7], en particulier à cause du quatrième[3] "V" régulièrement ajouté à la définition : celui de Valeur. En effet, les données sont régulièrement qualifiées d'or noir du $XXI^{\text{ième}}$ siècle.

Les verrous scientifiques autour des données massives sont non seuleument nombreux, mais en plus ténus et interconnectés. Cette thèse ne prétend nullement en lever, mais à peine effleurer celui du volume. Partant d'un jeu de données de taille raisonnable, les connaissances extraites sont souvent massives ; pour explorer ces connaissances, il faut les classifier et les ordonner en amont.

Prenons l'image d'un nouveau né : il est assailli par ses sens, il ne comprend pas ce qu'il voit, ce qu'il entend, ce qu'il touche, mais à force de structuration de ses sensations, il finit par donner du sens au monde qui l'entoure.

Une des clés des données massives réside sans doute dans la structuration ou l'organisation, au sens de classifier et ordonner. Attribuer une classe, aussi vite que possible, aux données, pour les regrouper en plus petits groupes homogènes, permet d'aborder globalement les trois "V". D'une certaine manière, il s'agit encore de diviser pour régner.

1.3 Organiser les données

La Recherche d'Informations et l'Extraction de Connaissances abordent la même problématique, mais sous deux angles différents : l'une ordonne (attribution

[2] Un jour quelqu'un m'a dit : "Pour toi, le Big Data, c'est juste 1Mo".

[3] Un cinquième "V" peut également être ajouté pour la Validité ou Véracité. Il fait référence à la qualité des données collectées.

d'un rang, du plus pertitent au moins), l'autre classifie (appartenance ou non à groupe). Les deux structurent et organisent les données.

La Recherche d'Information ordonne, à travers un système de classement, l'ordre est alors linéaire comme pour un moteur de recherche, ou à travers un sytème de hiérarchie, l'ordre est alors arborescent comme pour un système de fichiers. Cet ordonnancement, linéaire ou non, se fait suivant des critères déterminés par une requête. Entre en jeu alors, un autre point fondamental de la Recherche d'Informations : le langage de requêtes. Les bases de données ont SQL, très formalisé (syntaxe rigide), demandant un apprentissage, mais permettant la formulation de requêtes complexes. La recherche sur le Web tend vers le langage naturel, informel et facile d'accès mais pour lequel une requête complexe est parfois inaccessible. Entre les deux, "tout est possible".[4]

L'Extraction de Connaissances classifie les données, elle aussi de deux manières : par exemple en ajoutant des "étiquettes", les données peuvent alors avoir plusieurs "étiquettes" et donc appartenir à plusieurs groupes. Elle permet également la création de catégories, par exemple "spam" ou "non spam" pour les emails, "sain" ou "malade" pour des patients. Aucune donnée n'est alors présente dans plusieurs catégories.

Une fois ce processus terminé vient l'étape de visualisation pour l'analyste. Créer de "bonnes" et/ou "belles" visualisations est aussi important que difficile.

En complément de cette double organisation, ordonner et classifier, des processus de navigation plus ou moins guidée sont proposés afin d'affiner une requête, ou de préciser une classe.

Avec un pied dans la Recherche d'Informations et l'autre dans l'Extraction de Connaissances, l'Analyse de Concepts Formels [32, 173], via les treillis de concepts, ordonne et classe en même temps les données. Bref, elle structure et organise.

1.4 Analyse de concepts formels

L'analyse de concepts formels ("Formal Concept Analysis", FCA) a été introduite par R. Wille [181]. L'objectif annoncé de ce papier était de créer un rapprochement entre deux communautés. L'une étudiait la théorie des treillis et l'autre les utilisait pour structurer des données. La FCA repose sur les travaux antérieurs de M. Barbut et B. Monjardet [15], mais aussi sur toute la théorie des treillis [26, 41, 93]. Elle dispose également de solides fondements philosophiques issus de la logique de Port Royal [10]. L'approche initiale de la FCA consistait à restructurer la théorie des treillis pour faciliter ses applications. Une part importante de la FCA se situe également dans la logique, et un travail de restructuration de la logique a également été entrepris [189, 45].

La FCA trouve ses applications dans de nombreux champs : la Recherche d'Information, la Fouille de Données [58], la Représentation de la Connaissance, etc. Ces différents champs disposent chacun de nombreux outils reposant pour la plupart sur les statistiques et les probabilités. La FCA se démarque de ces

[4] "Anything is possible"

approches car elle adopte un point de vue symbolique/algébrique. Elle ne se concentre pas sur le contenu des données mais sur leur structure, apportant ainsi un point de vue complémentaire sur ces dernières.

L'analyse de concepts formels repose sur l'usage de données binaires : un objet, une observation possède ou non une propriété, un attribut. Dans sa forme originelle, la FCA n'utilise pas de données complexes. Néanmoins, il existe de nombreuses généralisations de la FCA qui permettent cela, par exemple les structures de motifs [5] (pattern structures).

Quoiqu'il en soit, l'entrée initiale de la FCA est un jeu de données binaires appelé contexte. Ce terme contexte est aussi à prendre au sens pratique : il s'agit du contexte de l'étude menée. Lorsque l'on étudie une situation ou un phénomène, comme la situation politique d'un pays, le bilan économique d'une entreprise, l'évolution d'une maladie, etc., il faut en premier lieu définir son contexte : ce qui fait partie de l'étude, les échanges avec ce qui en est extérieur, les informations ou connaissances que l'on a. D'un point de vue formel, ce contexte sera modélisé comme un jeu de données binaires.

Ensuite, la FCA en déduit un treillis de concepts. Cet objet étant de taille exponentielle, dans le pire des cas, en les données[6], il devient rapidement impossible de l'appréhender dans sa globalité, même si dans les cas pratiques, il peut rester de taille raisonnable. Ce treillis de concepts est l'objet principal de la FCA, il permet d'ordonner et classifier les données en même temps et cela de nombreuses manières. Un treillis de concepts apporte plusieurs ordres et classifications sur les données, apportant ainsi différents points de vues sur ces dernières.

D'une part, le treillis de concepts déduit d'un jeu de données binaire permet de faire une analyse très riche des données [7]. D'un autre côté, il est souvent d'une taille considérable. L'idée directrice de cette thèse est de décomposer : ou bien le jeu de données initial avec Hadoop, ou le treillis de concepts lui-même. Décomposer le jeu de données avec un outil fait pour les données massives permet d'attaquer le problème du "Big Input". Décomposer le treillis de concepts déduit d'un jeu de données permet là d'attaquer le problème du "Big Output". Le résultat d'une décomposition de treillis consiste en plusieurs petits treillis, de sorte qu'un processus itératif peut être mis en oeuvre.

1.5 Décompositions

L'idée principale d'une décomposition du treillis ou du contexte en plus petits éléments consiste à conserver les propriétés de classification du treillis initial.

[5] ou structure de patrons.

[6] Il s'agit d'une complexité théorique. En pratique, avec des données réelles, la taille est en général polynomiale.

[7] Les concepts permettent de créer des classes au sein des données. Ces classes peuvent éventuellement exclure des données, et certaines données peuvent éventuellement appartenir à plusieurs classes. D'autre part les concepts sont ordonnés (donc les données aussi indirectement) ; cet ordre n'est en général pas linéaire, donc plusieurs ordres linéaires sur les données peuvent être déduits.

Beaucoup de décompositions de treillis ont été définies et étudiées, à la fois du point de vue algébrique [51, 132] et du point de vue de la FCA [88, 83]. On peut citer la décomposition en produits directs [123] qui est une des plus simples, mais malheureusement seulement un petit nombre de treillis en admet une non-triviale, le théorème de factorisation [132], la décomposition matricielle [19], une décomposition basée sur la décomposition en valeurs singulières [84], la décomposition Atlas [88], la décomposition subtensorielle [88], la décomposition sous-directe [8] [51, 80, 77, 78, 179, 180, 182, 184, 83, 176], ou la construction de doublement de convexe inversée. Cette construction de doublement de convexe a elle-même été largement étudiée [46, 49, 139, 89, 24], principalement d'un point de vue théorique, pour caractériser les treillis qui peuvent être obtenus par cette construction. Basée sur la construction de doublement, nous proposons une nouvelle méthode intitulée "construction de doublement inversée" permettant de réduire la taille des données générées. En d'autres termes, nous proposons une méthode permettant de construire un treillis plus petit à partir d'un treillis donné en regroupant des concepts mis en relation par certaines relations de congruences. Cette construction est basée sur les travaux initiaux d'A. Day dont la méthode est connue sous le nom de construction de doublement de convexe [47, 46, 49]. Cette méthode a été généralisée [127, 89, 156] et largement étudiée [50, 139, 24]. Intuitivement, cette construction consiste à doubler un ensemble convexe C de noeuds à l'intérieur d'un treillis L.

La "construction de doublement inversée" consiste à retirer d'un treillis L un ensemble convexe de noeuds. Il est possible d'itérer cette méthode jusqu'à ce qu'il ne soit plus possible de retirer d'ensemble convexe. Cependant, il peut arriver que dès le départ, aucun convexe ne puisse être enlevé. Lorsque la construction de doublement inversée s'applique à un treillis supposé *fini* L et un convexe C, alors la construction de doublement d'A. Day permet de retrouver le treillis initial L ; ainsi, en conservant le treillis réduit et la mémoire du convexe, aucune information n'est perdue.

Les recherches sur la construction de doublement de convexe peuvent être organisées dans l'ordre chronologique :

- Les premières correspondent aux travaux originaux d'A. Day [46, 47, 48, 49, 50, 139], qui a introduit la construction. Au tout début, seuls des intervalles étaient doublés.
- Ensuite des généralisations ont été définies qui ont conduit à la version actuelle de la construction [89].
- En parallèle, des caractérisations de treillis pouvant être obtenus en itérant cette construction ont été explorées [47, 49, 50, 139, 24].
- Nous définissons une construction inversée qui permet d'enlever un convexe d'un treillis, lorsque cela est possible [175].

[8] Derrière cette traduction obscure, se cache l'idée simple de transformer un treillis en un produit direct de treillis et lorsque ce n'est pas possible, de seulement l'incarner un sous-objet d'un produit direct de treillis.

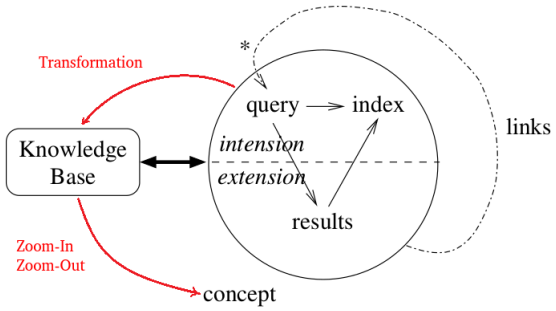


FIGURE 1.1 : LattExp vu comme une instance d'ACN enrichie d'une fonctionnalité de zoom.

Etre capable de retrouver tout le treillis à partir d'un plus petit treillis et d'un convexe contenu dedans, signifie que toute l'information est en fait contenue dans le petit treillis. Ainsi, nous avons seulement besoin de considérer le sous-contexte qui définit le petit treillis. En d'autres termes, seule une partie des données est pertinente ; cette approche peut donc être vue comme une méthode de sélection de caractéristiques (*feature selection*). Finalement, il suffit d'accéder et même de stocker seulement une plus petite partie des données. C'est évidemment intéressant dans un contexte de données massives omniprésent de nos jours dans beaucoup d'applications basées sur des données réelles.

Par ailleurs, d'autres approches que la décomposition permettent de contourner la taille du treillis, par exemple en générant seulement quelques concepts et leur voisinage de manière interactive [70, 177], ou à l'aide de meilleures visualisations (par exemple les diagramme imbriqués "nested line diagrams" [88]).

Il y a au moins deux raisons pour étudier ce cas de gestion de motifs extraits des données. La première vient du fait que les utilisateurs tendent à être débordés par le flot des ces motifs, même lorsqu'en entrée les données sont peu volumineuses. La seconde est que la FCA a considérablement progressé dans la construction et l'exploration des treillis et qu'il est possible maintenant d'adapter les solutions existantes pour cibler seulement les motifs utiles.

1.6 Le projet LattExp

Le projet LattExp, unifiant tout le contenu de ce document, peut être présenté presque comme une instance d'ACN (Navigation Conceptuelle Abstraite) (section 2.9).

Les points clés de LattExp sont les suivants :

- Basé sur la FCA, il permet de classifier, ordonner et visualiser les données sous forme d'un treillis de concepts. En cela, les élements essentiels d'une

instance d'ACN sont présents.

- Les décompositions intégrées à LattExp permettent une transformation des données qui génère de nouveaux concepts (essentiellement des regroupements de concepts existants) et donc de nouvelles visualisations. Concrètement, ces transformations apportent à l'utilisateur une fonctionnalité de zoom/dézoom interactif sur les visualisations.

Tous ces éléments sont résumés visuellement sur la figure 1.1.

1.7 Feuille de route

Au chapitre 2, nous donnons une brève introduction à la Recherche d'Informations (RI) et à l'Extraction de Connaissances (EC). Ensuite vient un état de l'art de l'Analyse de Concepts Formels (FCA). Puis ce chapitre est clos par une introduction à la Navigation Conceptuelle Abstraite (ACN) [70] qui est un cadre récent et puissant pour la recherche d'information et l'extraction de connaissances, basé sur la FCA.

Les chapitres 3, 4, 5, 6 regroupent les nouveautés. Le premier présente notre projet, LattExp. Ensuite vient un chapitre théorique présentant des décompositions de treillis et quelques résultats sur les implications. Le chapitre 5 est dédié à la présentation d'algorithmes pour la FCA utilisant Hadoop MapReduce. Enfin, cette partie est conclue par un chapitre d'expérimentations.

Les deux chapitres centraux de cette partie ont une approche opposée du problème de la volumétrie :

- En décomposant le treillis, on cherche à réduire le volume afin de pouvoir travailler avec des treillis de taille raisonnable. C'est une réduction de dimension. Certaines de ces méthodes de décomposition étant basées sur des sous-contextes, elles peuvent être également utilisées comme des index : les données sont réparties (indexées) suivant le sous-contexte auquel elles appartiennent dans la décomposition. Par ailleurs, d'autres stratégies d'indexation pour la FCA ont déjà été explorées [126]. Il s'agit là de traiter la dimension "Big Output".

- En utilisant Hadoop, on cherche cette fois à utiliser une architecture adaptée aux données massives dans le cadre de la FCA. L'approche est donc globale dans une infrastructure adaptée. Il s'agit là de traiter la dimension "Big Input".

Finalement, nous présentons les évolutions et les cas d'usage que nous souhaitons faire avec LattExp, puis donnons des perspectives et conclusions dans le chapitre 7.

Chapitre 2

Etat de l'art

Sommaire

2.1 Recherche d'information – Extraction de connaissances

Pour commencer un état de l'art relatif aux deux champs "Recherche d'Information" et "Extraction de Connaissances", il peut être utile, justement, d'utiliser les méthodes de ces deux champs, par exemple pour ordonner et/ou classifier les mot-clés suivants :

- Fouille de Données
- Apprentissage Automatique
- Analyse de Données
- Intelligence Artificielle
- Classification
- Visualisation
- ...

Considérons l'"Analyse de Données" dans son sens le plus large, alors elle devient le but à atteindre : nous sommes submergés par les données que nous souhaitons comprendre ou au moins utiliser. Pour cela, nous avons besoin d'outils. Dans la suite, nous allons nous concentrer sur la "Recherche d'Information" et l'"Extraction de Connaissances" qui sont deux champs fournissant des outils complémentaires pour l'Analyse de Données.

2.1.1 Recherche d'information – Ordonner puis Sélectionner

Commençons cette sous-section dédiée à la Recherche d'Information, par une citation du livre "Search Engines : Information Retrieval in Practice" [42] :

> Gerard Salton, a pioneer in information retrieval and one of the leading figures from the 1960s to the 1990s, proposed the following definition in his classic 1968 textbook (Salton, 1968) :
>
> Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information.

Il y a quelques étapes importantes dans un processus de recherche d'information [70, 42], cependant nous nous concentrerons sur seulement deux d'entre elles :

- **Langage de requêtes :** D'après S. Ferré [70], un langage de requêtes possède deux caractéristiques principales : la nature des questions qu'il autorise (de simples à complexes) et sa propre nature (de naturel à formel). Bien sûr, ces deux caractéristiques sont en partie déterminées par les données elle-mêmes. Des données bien structurées devraient permettre de facilement formuler des requêtes complexes en utilisant un langage plutôt formalisé, par exemple dans les RDBMS [1] ; à l'opposé, les données hétérogènes ne pourront sans doute être confrontées qu'à des requêtes relativement simples et donc avec un langage moins formel, par exemple dans les moteurs de recherche.

- **Classement :** Cette étape n'existe pas toujours. Le résultat d'une requête peut être seulement l'ensemble des données validant la requête. Cependant, lorsqu'elle existe, cette étape est "juste" [2] un classement. Cela signifie que la donnée la plus pertinente est donnée en premier, puis la suivante, etc. Un classement est un processus linéaire (il y a le premier, le deuxième, le troisième et ainsi de suite ; mais rien de plus), il n'est pas possible d'avoir des détails du genre "cette donnée possède ces qualités, et celle-là en possède d'autres". Ces détails ne peuvent être obtenus qu'avec des hiérarchies non-linéaires (i.e. des ordres partiels). D'autre part, classer n'est bien sûr pas "juste classer" dans le sens où il s'agit d'une étape extrêmement technique faisant intervenir des métriques et des algorithmes très complexes.

Dans une optique d'analyse de données, ces deux outils, i.e. les langages de requêtes et les méthodes de classement, fournis par le champ "Recherche d'Information", peuvent être utilisés comme des premières étapes : la sélection de données avec la requête et le prétraitement via le classement.

2.1.2 Extraction de connaissances – Classifier

L'Analyse de Données, du point de vue de la Recherche d'Information, est une analyse quantitative. D'un autre côté, cette analyse de données a également besoin d'outils qualitatifs qui fourniront une compréhension profonde des données. Cet aspect est abordé par le champ "Extraction de Connaissances". Un processus d'extraction de connaissances est en général décomposé en cinq étapes [6] :

- Sélection : A partir d'une bonne connaissance du domaine et des objectifs de l'extraction, une sélection est effectuée dans les données initiales.

- Prétraitement : Les données sont nettoyées ; les problèmes de valeurs manquantes ou abérrantes sont traités à cette étape.

- Transformation : par exemple, des méthodes de réduction de dimensions sont appliquées.

[1] *RDBMS : Relationnal DataBase Management Systems*
[2] Ce classement se faisant en suivant des critères qu'il est loin d'être évident de déterminer.

- Fouille de Données : par exemple la recherche de motifs particuliers, élaboration de fonctions de régression, etc.
- Interprétation / Evaluation : c'est l'étape finale, qui passe en particulier par des méthodes de visualisation.

Nous nous intéresserons principalement aux deux dernières étapes : la Fouille de Données et l'Interprétation dans laquelle les méthodes de visualisation sont fondamentales. Cependant, les méthodes de décompostion, vues comme méthodes de réduction de dimension et de traitement du volume, peuvent également être placées dans l'étape de transformation ou même de prétraitement. L'étape de Fouille de Données peut elle-même comporter les techniques suivantes :

- **La Fouille de Règles d'Associations :** Il s'agit de relations entre les différentes caractéristiques des données. La découverte de telles dépendances a beaucoup d'applications très variées, par exemple les recommandations d'achats conjoints de produits aux clients. Il s'agit donc d'une étape très importante dans la fouille de données, qui a été initiée par l'algorithme Apriori [5].
- **Clustering** [3] **:** Cela consiste à mettre des "étiquettes" nouvellement découvertes à un groupe homogène de données afin de les enrichir (en particulier au niveau sémantique), par exemple le profil d'un groupe de fraudeurs. La FCA pourrait être placée dans cette catégorie.
- **Classification** [4] **:** Il s'agit d'un processus d'apprentissage supervisé dans lequel il n'y a pas de recouvrement, i.e. chaque données possède une et une seule "étiquette" ; c'est donc un cas de partionnement des données. Par exemple les e-mails sont classés en "spam" ou "non-spam".

Ces trois techniques parmi plusieurs autres techniques de fouille de données, utilisent une très large variété de méthodes et d'outils, par exemple les statistiques, les réseaux de neurones, les séparateurs à vaste marge (SVM) [5], la réduction de dimensions.

Ces techniques tendent à être trop efficaces dans le sens où l'utilisateur est rapidement submergé par la connaissance calculée. Ainsi un bouclage peut être mis en place avec la Recherche d'Informations afin de sélectionner et classer cette connaissance. Ce bouclage apporte de l'interactivité à l'ensemble du processus. Evidemment, cette boucle doit être contrôlée par l'utilisateur et donc des outils de navigation, comme les graphes, les arbres, lui sont fournis. Ce processus de navigation est parfois couplé avec les outils de visualisation, par exemple avec la navigation par facettes (Nav. Fac.).

Pour résumer les caractéristiques des techniques impliquées dans ce processus, S. Ferré [70] propose le tableau 2.1.

[3] Peut se traduire en Segmentation (source FranceTerme), mais le terme n'est pas utilisé.

[4] En français, ce terme est parfois utilisé dans un autre sens que celui donné ici, en particulier cette description de la classification peut être considérée comme très restrictive.

[5] En anglais SVM pour Support Vector Machine, Machine à Vecteur de Support

Approche	expressivité	lisibilité	guidage	taille
Langages Formels	haute	difficile	non spécifique	BD
Langages Naturels (LN)	basse	naturelle	non spécifique	BD
LN Contrôlés	moyenne	facile	non spécifique	BD
Graphes	basse	naturelle	spécifique	Web
Hiérarchies	basse	naturelle	spécifique	Web
Treillis de Concepts	moyenne	facile	spécifique	BD
Nav. Fac.	moyenne	facile	spécifique	BD
OLAP	moyenne	facile	spécifique	BD

TABLE 2.1 : Comparaison des différentes approches d'accès à l'information. Consulter [70] pour les détails et justifications.

2.1.3 Ordonner et Classifier

Nous pouvons résumer ce double objectif (classifier et ordonner) à l'aide des considérations simples suivantes : pour analyser les données, nous avons besoin à la fois de les ordonner via les méthodes de Recherche d'Information, et de les classifier via les méthodes d'Extraction de Connaissances. Tout comme l'être humain qui se déplace sur ses deux pieds, l'analyste de données les traite à la fois avec les méthodes de Recherche d'Informations et d'Extraction de Connaissances. Dans la suite, nous utiliserons le terme "organiser" en référence à ce double processus d'ordonnancement [6] et de classification [7]. Ainsi, les différents éléments constitutifs d'un cadre complet pour l'analyse de données sont présentés dans la figure 2.1 ; il s'agit : d'un langage de requêtes, d'un procédé d'ordonnancement, d'une méthode de classification ou clustering, d'un outil de navigation interactive et d'un procédé de visualisation.

A l'exception du langage de requête, l'Analyse de Concepts Formels dispose de toutes ces caractéristiques.

2.2 Origines philosophiques de la FCA

Commençons par une citation de la préface de l'ouvrage de référence de B. Ganter et R. Wille[88] :

"L'analyse de concepts formels est un domaine des mathématiques appliquées basé sur la formalisation de la notion de concept [8] et de hiérarchie de concepts".

Ainsi, le premier objet manipulé par la FCA est le concept. Citons à nouveau R. Wille [188] à ce propos :

"Les concepts peuvent être compris, d'un point de vue philosophique, comme les atomes de la pensée."

On peut aller plus loin en résumant les titres des parties de ce papier [188] :

[6] Au sens d'ordonner avec une relation d'ordre.
[7] Au sens classification supervisée ou non-supervisée.
[8] Une définition formelle de concept sera donnée dans la sous-section 2.3.4

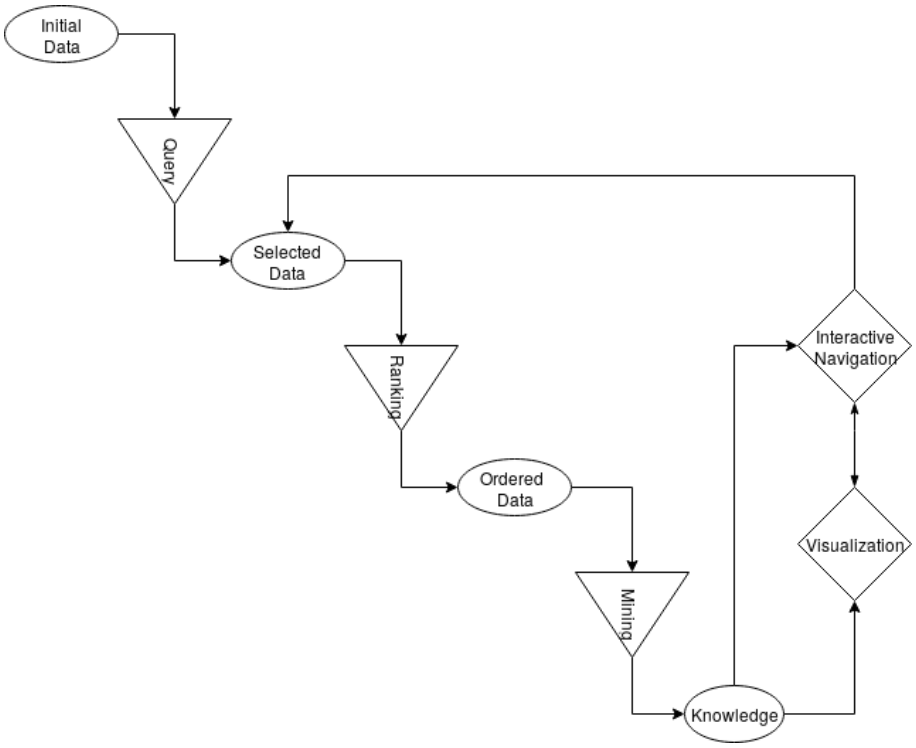


FIGURE 2.1 : Un cadre complet pour l'analyse de données.

- Les concepts sont des actes cognitifs et des unités de savoir.
- Les concepts, en tant qu'unité de savoir, se réfèrent au réel, ils sont analogues à des motifs de pensée.
- Le langage, en tant que système intermédiaire, permet/promeut l'actualisation des concepts.
- La formation des concepts n'est pas un automatisme formalisable, ils évoluent au cours du temps, en partie implicitement, et essentiellement à travers le langage.

La notion de concept est fondamentale car il s'agit de l'objet manipulé par la raison pour former des jugements et conclusions [189]. Ce point de vue se trouve d'ailleurs déjà dans "La logique de Port-Royal" [10]. V. Duquenne [61] souligne que "La logique de Port-Royal" contient une définition de concept en termes d'intension et d'extension, et même, cette notion de concept est utilisée pour modéliser la pensée humaine à l'aide de trois actions : concevoir, juger et raisonner. Plus précisément [10], on peut trouver dans "La logique de Port-Royal" les points suivant sur les concepts, la pensée humaine, et même les contextes et la logique :

- "Comme nous ne pouvons avoir aucune connaissance de ce qui est hors de nous, que par l'entremise des idées [concepts] qui sont en nous, les réflexions que l'on peut faire sur nos idées [concepts] sont peut-être ce qu'il y a de plus important dans la logique, parce que c'est le fondement de tout le reste."
- Les quatre parties de l'ouvrage sont organisées de la manière suivante : la première partie est à propos de ce que les auteurs appellent idées, notion qui s'avère très proche de celle de concept et liée à l'action de concevoir, la seconde partie traite des jugements, la troisième de raisonnements, et en particulier des règles permettant de mener un raisonnement valide. Les auteurs y définissent une taxonomie des syllogismes. En particulier, un grand soin est donné à la négation, dont on sait qu'elle peut être difficile à appréhender suivant que l'on fait l'hypothèse du monde ouvert ou du monde fermé. La dernière partie traite de méthode, ce qui, pour les auteurs, signifie démonstration, c'est à dire l'enchaînement des syllogismes.
- La citation suivante insiste sur l'importance du contexte : "Mais il faut remarquer que les mots sont généraux en deux manières : l'une, que l'on appelle univoque, qui est lorsqu'ils sont liés avec des idées générales ; [...] L'autre que l'on appelle équivoque, qui est lorsqu'un même son a été lié par les hommes à des idées différentes."

Cette représentation de la connaissance basée sur cette définition de concept a rarement été remise en cause [153] car c'est cette notion qui est à l'origine des multiples applications issues de la FCA.

Enfin, toute la FCA s'insère naturellement dans un cadre plus général de représentation de la connaissance [98].

2.3 Autour des treillis

L'analyse de concepts formels est basée sur la théorie mathématique des treillis et plus particulièrement les treillis de concepts. Ces derniers se révèlent être des outils particulièrement bien adaptés à la classification (les observations sont classées dans des concepts) et pour ordonner (les concepts sont organisés suivant une hiérarchie [9]). La classification repose essentiellement sur la notion de partition ou de relation d'équivalence. Dans le cas des treillis de concepts, les partitions n'apparaissent pas directement : les observations peuvent appartenir à plusieurs concepts ; de sorte que la FCA est plus proche du clustering (où il y a recouvrement, *overlapping* en anglais) que de la classification. Quant à l'ordonnancement, il repose évidemment sur la notion de relation d'ordre.

Relations d'équivalence et relations d'ordre sont toutes deux des cas particuliers de relations binaires. De même, un contexte formel est lui aussi une relation, plus générale. Enfin, les décompositions présentées ultérieurement, reposent également sur des relations : de congruence ou de tolérance. Ainsi toute la FCA et les développements présentés ici possèdent comme fondement commun la notion de relation, dont plusieurs cas particuliers différents sont utilisés à différents niveaux.

2.3.1 Relations binaires

Remarque : **Tous les ensembles considérés dans ce document sont supposés finis, en particulier les treillis.**

Généralement les relations binaires sont séparées en deux grandes familles : les relations d'ordre servant à comparer et les relations d'équivalence servant à classifier [15].

Définitions

Le terme « relation binaire » vient des mathématiques. Cependant, du point de vue de la FCA, il s'agit d'un cas particulier de contexte.

Définition 2.3.1. Etant donné un ensemble E, une relation binaire sur E est la donnée d'une partie R de $E \times E$.

Remarques : Nous ne considèrerons que des ensembles finis, même si dans certaines situations cette restriction n'est pas nécessaire.

Lorsque $(e_1, e_2) \in R$, il est d'usage de noter $e_1 R e_2$ et cela signifie que e_1 est en relation avec e_2.

Exemple 2.3.1. Considérons sur l'ensemble $E = \{1, 2, 3\}$, la relation binaire $R = \{(1, 1), (2, 2), (3, 3)\}$.

Cette relation peut être visualisée avec le tableau 2.2 .

Deux éléments sont en relation si et seulement s'ils sont égaux. Ainsi, avec la notation $e_1 R e_2$, on constate que cette relation est $e_1 = e_2$.

[9] i.e. un ordre partiel.

TABLE 2.2 : Relation d'égalité

	1	2	3
1	x		
2		x	
3			x

Définition 2.3.2. Soit E un ensemble et R une relation binaire sur cet ensemble. On dit que :

(i) R est réflexive lorsque $\forall x \in E,\ xRx$

(ii) R est symétrique lorsque $\forall x, y \in E,\ xRy \Longrightarrow yRx$

(iii) R est anti-symétrique lorsque $\forall x, y \in E,\ xRy \,\&\, yRx \Longrightarrow x = y$

(iv) R est transitive lorsque $\forall x, y, z \in E,\ xRy \,\&\, yRz \Longrightarrow xRz$

Remarque : La seule relation binaire possédant ces quatre propriétés est la relation d'égalité avec (ii) et (iii).

Deux grandes familles de relations binaires joueront une place centrale dans ce document : les relations d'ordre et les relations d'équivalence. Ces deux familles ont peut-être d'ailleurs inspiré le titre de l'ouvrage de M. Barbut et B. Monjardet [15].

Relations d'ordre

Définition 2.3.3. Une relation binaire qui est reflexive, anti-symétrique et transitive est appelée relation d'ordre. Un ensemble muni d'une relation d'ordre est appelé ensemble ordonné.

Remarques : On utilisera la notation $x \leq y$ au lieu de xRy dans le cas des relations d'ordre. Lorsque $x \leq y$, on dit que x est un prédécesseur de y et que y est un successeur de x.

De plus, on notera $x < y$ lorsque $x \leq y$ et $x \neq y$.

Il s'agit d'une notion d'ordre partiel, c'est à dire qu'il est tout à fait possible que deux éléments ne soient pas comparables (notés $x \parallel y$). Dans le cas particulier où deux éléments sont toujours comparables, on dit que l'ordre est total.

Exemple 2.3.2. Considérons la figure 2.2 des douze couleurs. Le *Black* est plus grand que toutes les autres couleurs, alors que $White$ est plus petit. Nous avons également $Pink \leq Red \leq Marron$. Les couleurs *Red* et *Lime* ne sont pas comparables.

Une notion fondamentale découlant directement de la définition de relation d'ordre est celle de successeur/prédécesseur immédiat.

Définition 2.3.4. Soit $(E, \leq)$ un ensemble ordonné et $x \in E$ un élément de E. On dit que y est un successeur immédiat de x lorsque $x < y$, et qu'il n'y a pas de $z \in E$ tel que $x < z < y$.

Si y est un successeur immédiat de x, alors x est un prédécesseur immédiat de y.

Exemple 2.3.3. A nouveau avec l'exemple 2.2 des couleurs, les éléments du treillis *Red* et *Olive* sont des successeurs immédiat de *Orange*.

Les ensembles ordonnés sont couramment représentés via leur diagramme de Hasse :

Définition 2.3.5. Le diagramme de Hasse d'un ensemble ordonné est un graphe dont les noeuds sont les éléments de l'ensemble, puis un segment, joint x à y lorsque x est un successeur immédiat de y, de haut en bas [10].

Exemple 2.3.4. Le diagramme de Hasse de l'ensemble des couleurs est donné sur la figure 2.2.

Relations d'équivalence

Définition 2.3.6. Une relation binaire qui est reflexive, symétrique et transitive est appelée relation d'équivalence.

Remarque : Nous allons voir très rapidement que l'ensemble des relations d'équivalence sur un ensemble E est en bijection avec l'ensemble des partitions de E. De ce point de vue, les relations d'équivalence servent à fabriquer des classes.
Ces relations d'équivalence ont déjà été largement étudiées dans des travaux anciens [145]. Il s'agit d'une notion bien maitrisée.

Exemple 2.3.5. Un exemple classique est le suivant :
on considère un troupeau de moutons. On note E l'ensemble des pattes des moutons et l'on met sur E la relation binaire suivante : deux pattes sont en relation si et seulement si elles appartiennent au même mouton. Cette relation est une relation d'équivalence.

Le lien entre relation d'équivalence et partition se fait à travers la notion de classe d'équivalence :

Définition 2.3.7. Etant donnés un ensemble E et une relation d'équivalence R sur E, la classe d'équivalence d'un élément x de E est :

$$\bar{x} = \{y \in E,\, xRy\}$$

Exemple 2.3.6. Dans l'exemple précédent du troupeau de moutons, la classe d'une patte est l'ensemble des pattes du même mouton.

[10] Suivant les communautés, soit les grands éléments sont au-dessus des petits, soit au dessous.

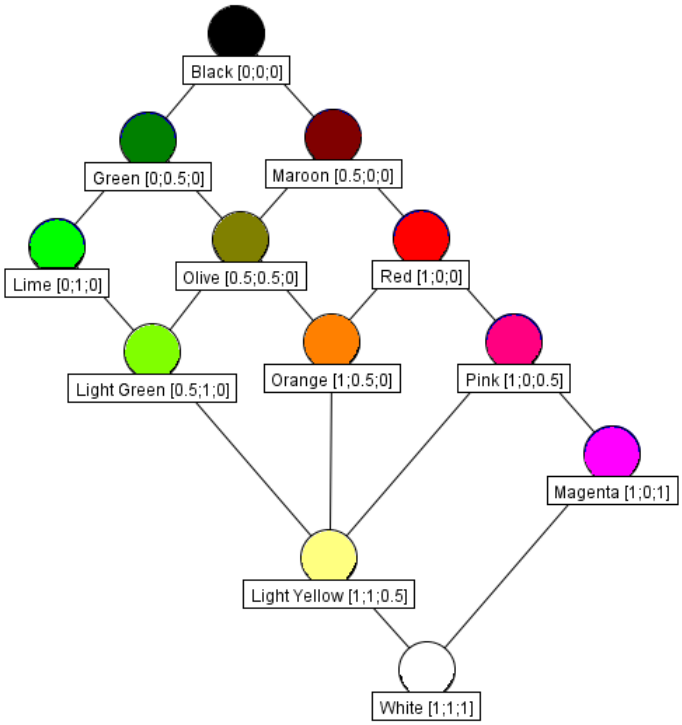


FIGURE 2.2 : Diagramme de Hasse de l'ensemble des couleurs.

Enfin nous avons :

Proposition 2.3.1. *L'ensemble des classes d'équivalence forme une partition.*

Remarque : Réciproquement, il est possible de construire une relation d'équivalence à partir d'une partition.

Exemple 2.3.7. Toujours dans l'exemple des moutons, la partition des pattes est justement donnée par les moutons qui sont les classes d'équivalence : toute patte appartient à un et un seul mouton.

2.3.2 Eléments particuliers

On se donne un ensemble ordonné fini $(E, \leq)$ et l'on va s'intéresser à des éléments particuliers de cet ensemble.

Définition 2.3.8. Soit $X \subseteq E$ une partie de $(E, \leq)$.

- Un élément $m \in E$ est un minorant de X lorsque m est plus petit que tous les éléments de X ; formellement :

$$\forall x \in X,\, m \leq x$$

- Un élément $M \in E$ est un majorant de X lorsque M est plus grand que tous les éléments de X ; formellement :

$$\forall x \in X,\, x \leq M$$

Remarque : Il est possible qu'un majorant(minorant) appartienne ou non à l'ensemble. Cette distinction influera sur les définitions suivantes.

Exemple 2.3.8. Dans l'exemple 2.2 des couleurs, avec $X = \{Olive, Red\}$, alors d'une part $Maroon$ et $Black$ sont des majorants et d'autre part $Orange$ et $White$ sont des minorants.

Définition 2.3.9. Soit $X \subseteq E$ une partie de $(E, \leq)$.

- Un élément $x \in E$ est un plus petit élément de X si $x \in X$ et :

$$\forall y \in X, x \leq y$$

- Un élément $x \in E$ est un plus grand élément de X si $x \in X$ et $\forall y \in X, y \leq x$.

Remarques : Lorsqu'il existe, un plus petit (grand) élément est unique.

Le plus grand élément est un majorant de l'ensemble qui est dans l'ensemble. Il est également comparable à tous les éléments de l'ensemble.

Dans le cas particulier où la partie X considérée est l'ensemble E tout entier, le plus petit élément (lorsqu'il existe) sera noté $\perp$ et le plus grand élément (lorsqu'il existe) sera noté $\top$.

Exemple 2.3.9. Dans l'exemple 2.2 des couleurs, $Black$ est la plus grande des onze couleurs alors que $White$ est la plus petite.

La définition suivante est la plus importante en ce qui concerne les treillis et nécessite les deux définitions précédentes.

Définition 2.3.10. Soit $X \subset E$ une partie de $(E, \leq)$.

- Lorsqu'il existe, le plus petit des majorants de X est appelé borne supérieure ou supremum (*least upper bound, lub* en anglais) de X et notée $\sup(X)$ ou $\vee X$.
- Lorsqu'il existe, le plus grand des minorants de X est appelé borne inférieure ou infimum (*greatest lower bound, glb* en anglais) de X et notée $\inf(X)$ ou $\wedge X$.

On remarquera que l'infimum (supremum) n'est pas nécessairement dans la partie considérée.

Exemple 2.3.10. Dans l'exemple des couleurs 2.2, avec $X = \{Olive, Red\}$, alors $Maroon$ est le supremum de X et $Orange$ est l'infimum de X.

Avant de passer aux treillis, donnons une dernière définition qui sera utile lors de l'utilisation des convexes.

Définition 2.3.11. Soit $X \subseteq E$ une partie de $(E, \leq)$.

- Un élément maximal de X est un élément M de X tel que X ne possède pas d'élément plus grand que M ; formellement :

$$M \in X \,\&\, \forall x \in X,\ M \leq x \Longrightarrow x = M$$

- Un élément minimal de X est un élément m de X tel que X ne possède pas d'élément plus petit que m ; formellement :

$$m \in X \,\&\, \forall x \in X,\ x \leq m \Longrightarrow x = m$$

Remarques : Un ensemble peut avoir plusieurs éléments maximaux (minimaux).

Un élément maximal (minimal) est toujours dans la partie considérée.

Lorsqu'un ensemble admet un plus grand élément, alors c'est nécessairement aussi son unique élément maximal. Mais un élément maximal n'est pas forcément le plus grand élément car il est possible qu'il ne soit pas comparable à certains éléments de l'ensemble (s'il n'est pas unique).

Exemple 2.3.11. Dans l'exemple des couleurs 2.2, considérons l'ensemble X :

$$X = \{Olive, Orange, Red, Pink\}$$

Alors d'une part, les éléments maximaux sont $Olive$ et Red, d'autre part les éléments minimaux sont $Orange$ et $Pink$. On remarque que Red et $Olive$ ne sont pas comparables entre eux et donc ne sont ni l'un ni l'autre plus grand élément.

2.3.3 Treillis

Treillis

Pour le moment nous donnons uniquement la définition formelle ; la structure de treillis est à la base de la FCA, les treillis seront présents tout au long de ce manuscrit dans diverses situations concrètes.

Définition 2.3.12. Un treillis est un ensemble ordonné dans lequel toute paire (x, y) d'éléments admet une borne supérieure notée $x \vee y$ et une borne inférieure $x \wedge y$.

Remarque : Un treillis complet est tel que toute famille d'éléments possède une borne supérieure et une borne inférieure. Comme nous ne considérons que le cas fini, tous nos treillis sont complets.

Exemple 2.3.12. L'ensemble des parties $\mathcal{P}(M)$ d'un ensemble M muni de l'inclusion est un treillis, le supremum étant l'union, l'infimum étant l'intersection.

L'ensemble des partitions d'un ensemble (ou des relations d'équivalences sur cet ensemble) est naturellement muni d'une structure de treillis, où l'ordre est l'inclusion des graphes. Ce treillis a été abondamment étudié [64].

Eléments irréductibles

Définition 2.3.13. Un élément $j \neq \bot$ est dit sup-irréductible lorsqu'il n'est pas borne supérieure d'éléments distincts de lui-même.

Théorème 2.3.2. *[88] Un élément j d'un treillis est sup-irréductible si et seulement s'il admet un unique prédécesseur immédiat, noté j^-.*

Définition 2.3.14. Un élément $m \neq \top$ est dit inf-irréductible lorsqu'il n'est pas borne inférieure d'éléments distincts de lui-même.

Théorème 2.3.3. *[88] Un élément m d'un treillis est inf-irréductible si et seulement s'il admet un unique successeur immédiat, noté m^+.*

Exemple 2.3.13. A nouveau avec l'exemple 2.2 des couleurs, $Green$, $Maroon$, $Lime$, Red, $Pink$ et $Magenta$ sont inf-irréductibles.
D'autre part, $LightYellow$, $Magenta$, $LightGreen$, $Orange$, et $Green$ sont sup-irréductibles.

Remarques : Les successeurs immédiats de $\bot$, sont sup-irréductibles et sont appelés atomes.

Dualement, les prédécesseurs immédiats de $\top$, sont inf-irréductibles et sont appelés co-atomes.

Filtres et Idéaux

Bien que peu utilisés [11] dans ce document, les notions d'idéal et de filtre sont très importantes dans la théorie des treillis :

[11] mais pas inutile

Définition 2.3.15. Un intervalle $[x, y]$ d'un treillis L est défini par :

$$[x, y] = \{z \in L,\ x \leq z \leq y\}$$

Un idéal I d'un treillis L est un intervalle de la forme $[\bot, x]$. Un filtre I d'un treillis L est un intervalle de la forme $[x, \top]$, et est la notion duale (i.e. en reversant l'ordre) de celle d'idéal.

Remarque : Rappelons que tous les ensembles considérés dans ce document sont finis, en particulier les treillis. La définition précédente est adaptée à cette situation.

2.3.4 Treillis de concepts ou de Galois

Définition 2.3.16. Un contexte formel (O, A, R) est défini par un ensemble O d'objets, un ensemble A d'attributs, et une relation binaire $R \subset O \times A$, entre O et A.

Remarque : Notons qu'il y a ici une très légère généralisation, car la relation binaire R se fait entre deux ensembles distincts (O et A), ce qui ne correspond pas à la définition 2.3.1 où il n'y avait qu'un seul ensemble E.

On en déduit deux opérateurs :

- Pour toute partie $X \subseteq O$, on pose : $X' = \{m \in A,\ \forall j \in X\ j\, R\, m\}$ et dualement,
- pour toute partie $Y \subseteq A$, on pose : $Y' = \{j \in O,\ \forall m \in Y\ j\, R\, m\}$.

Exemple 2.3.14. Le tableau 2.3 est un exemple de contexte formel où une croix dans une cellule (j, m) signifie que l'objet j possède l'attribut m.

Définition 2.3.17. Un concept (formel) est un couple (X, Y), avec X une partie de l'ensemble des objets, Y une partie de l'ensemble des attributs, tel que $X' = Y$ et $Y' = X$. Les ensembles X et Y sont respectivement appelé extension et intension[12] du concept.

Remarques : Les concepts (X, Y) sont aussi les sous rectangles maximaux $X \times Y \subseteq R$.

Les applications $.' : \mathcal{P}(O) \to \mathcal{P}(A)$ et $.' : \mathcal{P}(A) \to \mathcal{P}(O)$ sont décroissantes, c'est à dire $T_1 \subseteq T_2 \implies T_2' \subseteq T_1'$. Les applications $.'' : \mathcal{P}(O) \to \mathcal{P}(O)$ et $.'' : \mathcal{P}(A) \to \mathcal{P}(A)$ sont extensives, c'est à dire $T \subseteq T''$. Ces propriétés font du couple d'applications $(.', .')$ une *correspondance de Galois* [13].

L'ensemble des concepts déduit d'un contexte peut être ordonné par :

$$(X_1, Y_1) \leq (X_2, Y_2) \iff X_1 \subseteq X_2 \iff Y_2 \subseteq Y_1 \tag{2.1}$$

[12] La sémantique du terme intention est bien différente.

[13] Une correspondance de Galois est un couple d'applications décroissantes dont les composées sont extensives.

	R-0.5	G-0.5	R-1	G-1	B-0.5	B-1
Light Yellow	x	x	x	x	x	
Magenta	x		x		x	x
Orange	x	x	x			
Light Green	x	x		x		
Lime		x		x		
Black						
White	x	x	x	x	x	x
Pink	x		x		x	
Olive	x	x				
Red	x		x			
Green		x				
Maroon	x					

TABLE 2.3 : Un exemple de contexte. Les couleurs sont codées par un réel entre 0 et 1, sur les trois canaux Red-Green-Blue. Les couleurs de cet exemple n'ont utilisé que trois valeurs 0 (pour une absence), 0.5 et 1.

Définition 2.3.18. L'ensemble de tous les concepts muni de cette relation d'ordre est un treillis complet, appelé treillis de concepts ou treillis de Galois du contexte (O, A, R).

Définition 2.3.19. Différents contextes peuvent donner des treillis de concepts isomorphes. Parmi ces contextes, il en existe un unique (à isomorphisme près), appelé contexte réduit défini par deux ensembles O et A de taille minimale.

Un représentant particulier est formé à l'aide des éléments irréductibles du treillis (dans ce cas $O = J_L$ et $A = M_L$).

Exemple 2.3.15. La figure 2.3 montre le treillis de concepts déduit du contexte 2.3. Les noeuds avec une étiquette au-dessus sont appelés concept-attributs et sont de la forme (a', a'') où a est un attribut. Les noeuds avec une étiquette au-dessous sont les concept-objets et sont de la forme (o'', o') où o est un objet.

Un résultat fondamental [15] établit que tout treillis $(L, \leq)$ est isomorphe au treillis de concepts du contexte $(J_L, M_L, \leq)$, où J_L et M_L sont respectivement les sup et inf irréducibles de L. De plus, ce contexte est réduit.

Le corollaire direct est que l'on peut passer (dans les deux sens) d'un treillis et au contexte réduit où les objets du contexte sont les sup-irréductibles du treillis et les attributs de contexte sont les inf-irréductibles.

Exemple 2.3.16. En ne conservant que les cinq premières lignes du contexte 2.3, on obtient le contexte réduit du treillis de la figure 2.3.

Lorsque cela sera utile, les operateurs $.'$ introduits précédemment seront

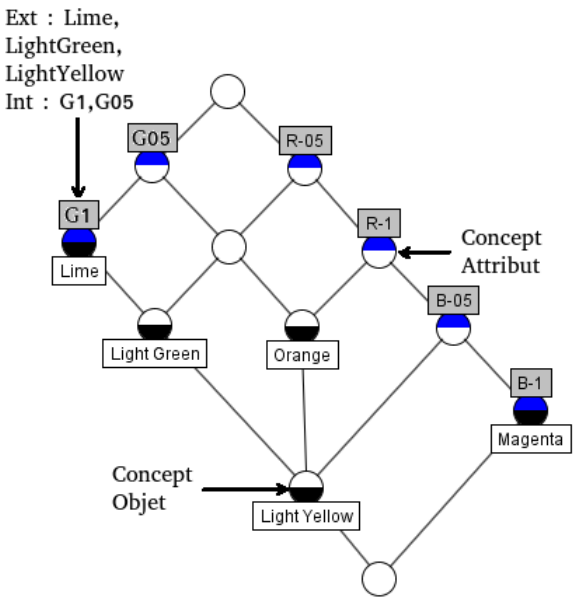


FIGURE 2.3 : Le treillis de concepts du contexte 2.3

notés $.^L$ pour souligner qu'ils sont utilisés avec le contexte réduit du treillis L ou $.^R$ pour souligner leur dépendance avec la relation R [14].

Ajoutons également quelques termes relatifs aux contextes :

Définition 2.3.20.

- Un sous-contexte consiste à restreindre la relation binaire à un sous-ensemble d'objets et d'attributs du contexte initial, formellement :

  Définition 2.3.21. Soit (O_1, A_1, R) un contexte. Un sous-contexte est un contexte de la forme $(O_2, A_2, R \cap (O_2 \times A_2)$ avec $(O_2 \subseteq O_1$ et $(A_2 \subseteq A_1)$.

- Le contexte complémentaire d'un contexte (O, A, R) est défini par :

$$(O, A, O \times A \backslash R)$$

- Un contexte multivalué est un contexte pour lequel les valeurs ne sont plus seulement booléennes. Il existe beaucoup de méthodes permettant de se ramener à un contexte binaire.

Terminons cette introduction des treillis de concepts par quelques remarques :

- Le treillis est une forme de classification (ou plus précisément de clustering), plus élaborée qu'une simple partition ou une simple hiérarchie.
- Le treillis est un classifiant non-supervisé. Dans le cas le plus simple, il peut être utilisé sous la forme : être ou ne pas être dans l'extension d'un concept.
- D'un point de vue de l'"Extraction de Connaissances", un résultat de fouille de données fourni par la FCA prend la forme d'un concept ou d'une règle d'association voire même d'un treillis.

2.3.5 Treillis de fermés

Du point de vue de la FCA, la donnée de départ est un contexte binaire duquel on déduit un treillis de concepts. Le théorème fondamental de la FCA [15, 88] nous assure que réciproquement, tout treillis fini est isomorphe au treillis de concepts de son contexte réduit.

Un troisième point de vue est couramment utilisé, celui des systèmes de fermeture ou opérateurs de fermeture [146, 33, 34].

Définition 2.3.22. Soit $(E, \leq)$ un ensemble ordonné. Un opérateur de fermeture φ sur E est une application $\varphi : E \to E$ telle que :

[14] Lorsque deux relations différentes R et S seront utilisées, cette notation sera indispensable pour préciser par rapport à quelle relation la dérivation est calculée.

- $\forall x \in E$, $x \leq \varphi(x)$, φ est extensive.
- $\forall x, y \in E$, $x \leq y \Longrightarrow \varphi(x) \leq \varphi(y)$, φ est croissante.
- $\forall x \in E$, $\varphi(x) = \varphi(\varphi(x))$, φ est idempotente.

Un fermé est un élément de l'image de φ.

D'un contexte binaire $K = (O, A, R)$, on déduit aisément un opérateur de fermeture sur 2^A, ensemble des parties de A, défini par $X \mapsto X''$ et dont le treillis des fermés est anti-isomorphe au treillis de concepts. De même, d'une base d'implications (voir section 2.4), on déduit un opérateur de fermeture. Les trois points de vue (contexte, treillis, opérateur de fermeture) sont équivalents (voir par exemple [20, 21] même si le résultat est antérieur) ; le dernier permet de faire le lien avec la topologie. Cependant, il s'agit d'une topologie finie donc en général laissée de côté.

Par ailleurs, une problématique classique de la FCA est de travailler efficacement avec des données non-binaires ; plusieurs généralisations de la FCA existent (Section 2.7) pour cela ; l'une d'elles utilise des opérateurs de fermeture [148]. Ces opérateurs de fermeture sont nombreux, par exemple l'union, l'intersection, etc [15]. Même dans le cas binaire, en considérant également l'absence d'attributs ou les attributs négatifs, on peut obtenir différents opérateurs de fermeture intéressants [158].

2.3.6 Les relations flèches

Exemple 2.3.17. Considérons à nouveau le contexte de la table 2.3.

Lorsqu'il y a une croix entre une observation et un attribut, on sait qu'ils sont en relation. L'absence de croix n'apporte rien ; il est cependant possible de combler ces vides dans le contexte en introduisant des liens plus subtiles, les relations flèches.

Considérons le contexte réduit $(J_L, M_L, \leq)$ d'un treillis $(L, \leq)$. Les relations flèches et le cercle [41, 91] forment une partition de la relation $\not\leq$ (definie comme le complémentaire de $x \leq y$) en considérant le prédécesseur immédiat j^- d'un sup-irréductible j, et l'unique successeur immédiat m^+ d'un inf-irréductible m :

- $j \updownarrow m$ si $j \not\leq m$, $j \leq m^+$ et $j^- \leq m$.
- $j \uparrow m$ si $j \not\leq m$, $j \leq m^+$ et $j^- \not\leq m$.
- $j \downarrow m$ si $j \not\leq m$, $j \not\leq m^+$ et $j^- \leq m$.
- $j \circ m$ si $j \not\leq m$, $j \not\leq m^+$ et $j^- \not\leq m$.

[15] Par exemple, si les attributs sont des intervalles, la fermeture d'une famille d'intervalle peut être définie comme l'intersection de ces intervalles.

	R-0.5	G-0.5	R-1	G-1	B-0.5	B-1
Light Yellow	×	×	×	×	×	↕
Magenta	×	↕	×	↓	×	×
Orange	×	×	×	↕	↕	∘
Light Green	×	×	↕	×	↓	∘
Lime	↕	×	∘	×	∘	∘

TABLE 2.4 : Le contexte réduit du treillis de la figure 2.3 complété avec les flèches et le cercle.

Le contexte réduit de la figure 2.3 est complété avec les quatre relations $\updownarrow$, $\uparrow$, $\downarrow$, et $\circ$ dans les cases vides correspondant aux cas où $j \not\leq m$:

Par exemple, on peut prendre $j = Magenta$ et $m = V - 0.5$ comme sup-irréductible et inf-irréductible respectivement (voir figure 2.3). Alors, $j^- = \perp$ et $m^+ = \top$. La relation $Magenta \updownarrow V - 0.5$ a lieu dans la table 2.3 car $Magenta \not\leq V - 0.5$, $\perp \leq V - 0.5$ et $Magenta \leq \top$.

2.3.7 Sous-contextes compatibles et flèche-fermés.

Cette sous-section est consacrée à l'équivalence entre les sous-contextes compatibles et les sous-contextes flèche-fermés.

A partir de cette section, des sous-contextes interviennent. Le contexte initial sera noté (O, A, R) et son treillis de concepts L. Les ensembles d'irréductibles de L seront notés $J_L \subseteq O$ et $M_L \subseteq A$. Lorsqu'un sous-contexte sera utilisé, il sera noté $(J, M, R \cap J \times M)$. Les parties $J \subseteq O$ et $M \subseteq A$, non-indicées, sont alors quelconques.

Sous-contextes Compatibles

Définition 2.3.23. Un sous-contexte $(J, M, R \cap J \times M)$ de (O, A, R) avec $J \subseteq O$ et $M \subseteq A$ est dit compatible si pour chaque concept (H, N) de (O, A, R), $(J \cap H, M \cap N)$ est un concept de $(J, M, R \cap J \times M)$.

Sous-contextes flèche-fermés

Définition 2.3.24. Un sous-contexte $(J, M, R \cap J \times M)$ d'un contexte (O, A, R) est un sous-contexte flèche-fermé lorsqu'il satisfait les conditions suivantes :

- Si ($j \uparrow m$ ou $j \updownarrow m$) et $j \in J$ alors $m \in M$
- Si ($j \downarrow m$ ou $j \updownarrow m$) et $m \in M$ alors $j \in J$

Théorème d'équivalence

Enonçons maintenant le théorème d'équivalence, dont la preuve se trouve dans [88] :

Théorème 2.3.4. *Soit $(J, M, R \cap J \times M)$ un sous-contexte de (O, A, R). Les propositions suivantes sont équivalentes :*

- *Le sous-contexte $(J, M, R \cap J \times M)$ est compatible.*
- *Le sous-contexte $(J, M, R \cap J \times M)$ est flèche-fermé.*

2.3.8 Relations de congruence et treillis quotient

Dans cette sous-section, nous introduisons l'équivalence entre les relations de congruence et les sous-contextes flèche-fermés.

Quotient

Soit R une relation d'équivalence sur un ensemble E.

Rappelons que la classe d'équivalence de $x \in E$ est :

$$x_R = \{y \in E \,|\, xRy\} \tag{2.2}$$

et que l'ensemble des classes d'équivalence, appelé ensemble quotient E/R, est :

$$E/R = \{x_R \,|\, x \in E\} \tag{2.3}$$

Treillis quotient

Définition 2.3.25. Une relation de congruence Θ sur un treillis L est une relation d'équivalence compatible avec les lois du treillis, c'est à dire telle que :

$$x_1 \Theta y_1 \text{ et } x_2 \Theta y_2 \Longrightarrow x_1 \wedge x_2 \Theta y_1 \wedge y_2 \tag{2.4}$$

$$x_1 \Theta y_1 \text{ et } x_2 \Theta y_2 \Longrightarrow x_1 \vee x_2 \Theta y_1 \vee y_2 \tag{2.5}$$

On met sur le quotient L/Θ la relation d'ordre suivante :

$$x_\Theta \leq y_\Theta \Longleftrightarrow x \Theta (x \wedge y) \Longleftrightarrow (x \vee y) \Theta y \tag{2.6}$$

Muni de cet ordre, L/Θ est un treillis, appelé treillis quotient. Un théorème standard de l'algèbre générale, dont la preuve est omise, affirme :

Théorème 2.3.5. *La projection $L \to L/\Theta$ est un morphisme de treillis surjectif.*

Exemple 2.3.18. Nous continuons avec le même exemple. Sur la partie gauche de la figure 2.4 est représentée une partition des noeuds définissant une relation de congruence. Sur la partie droite, le treillis quotient correspondant est donné.

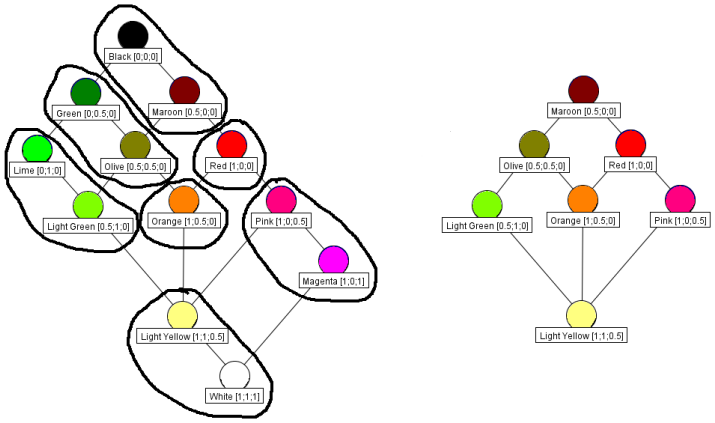


FIGURE 2.4 : Sur la partie gauche de la figure est représentée une partition des noeuds définissant une relation de congruence. Sur la partie droite, le treillis quotient correspondant est donné.

Théorème d'équivalence

Nous pouvons maintenant formuler le théorème suivant dont la preuve se trouve dans [88] :

Théorème 2.3.6. *Etant donné un treillis L, l'ensemble des relations de congruences sur L correspond bijectivement à l'ensemble des sous-contextes flèche-fermés du contexte réduit de L.*

Les relations de congruence seront calculées en utilisant ce théorème. Cependant, d'autres algorithmes existent : la méthode de Dilworth [41], la méthode de Grätzer [92], la méthode liée à la relation D [48] et les méthodes générales de R. Freese [77, 78, 80].

2.3.9 Relations de tolérance

Définition 2.3.26. Une relation binaire R sur un ensemble E qui est réflexive et symétrique est une relation de tolérance.

Remarques : Une relation d'équivalence est une relation de tolérance transitive. Ainsi les relations de tolérance sont plus nombreuses.

Les blocs sont aux tolérances ce que les classes d'équivalence sont aux relations d'équivalence. L'ensemble des blocs ne forme pas une partition, mais seulement un recouvrement ; les intersections peuvent être non vides.

Intuitivement, les relations de tolérances correspondent à l'idée de proximité ou de voisinage. Un élément est proche de lui-même (réflexivité), si x est proche

de y alors y est proche de x (symétrie), mais x peut être proche de y, y proche de z sans que x et z ne soient proches l'un de l'autre (absence de transitivité).

Définition 2.3.27. Une relation de tolérance complète T sur un treillis L est une relation de tolérance telle que :

$$x_1 T y_1 \text{ et } x_2 T y_2 \Longrightarrow x_1 \wedge x_2 T y_1 \wedge y_2 \tag{2.7}$$

$$x_1 T y_1 \text{ et } x_2 T y_2 \Longrightarrow x_1 \vee x_2 T y_1 \vee y_2 \tag{2.8}$$

Remarque : La seule différence entre une relation de congruence et une relation de tolérance complète est l'axiome de transitivité. Une relation de tolérance complète est une relation de tolérance qui est compatible avec la structure du treillis. Dans la suite, nous ne considèrerons que des relations de tolérance complète, donc le terme "complète" sera omis.

Les relations de tolérance ont été beaucoup étudiées par le passé, en particulier par I. Chajda and B. Zelinka ; de nombreux résultats sont connus à leur égard mais aussi au sujet du treillis des tolérances [14, 36, 35, 141, 142].

Dans le but de réduire la taille d'un treillis de concepts, il est possible de passer au quotient en utilisant non seulement une relation de congruence mais également en utilisant une relation de tolérance [43]

Avec une autre approche de la décomposition, les relations de tolérance sont également importantes dans le cadre de la décomposition Atlas [88, 94], qui, comme un atlas du monde, donne des parties de treillis qui se recouvrent.

Dans le cadre de l'étude des propriétés de relèvement dans les treillis, il est important de noter qu'il a été prouvé [44] que toute tolérance se relève en une congruence [16]. Cela signifie que si T est une relation de tolérance sur L, alors il existe un treillis K, une congruence C sur K et un morphisme de treillis $\varphi : K \to L$ tels que $\varphi(C) = T$.

Donnons quelques détails supplémentaires sur les relations de tolérances dont nous aurons besoin plus tard.

Définition 2.3.28. Etant donnés une relation de tolérance T sur un treillis L et $a \in L$, le bloc de a est :

$$[a] = \{x \in L,\, xTa\}$$

Remarque : Les blocs sont aux tolérances ce que les classes sont aux congruences.

A propos des blocs, nous avons la propriété suivante [88] :

Proposition 2.3.7. *Tout bloc* $[a]$ *est un intervalle* $[a] = [a_T, a^T]$*, avec :*

$$a_T = \wedge\{x,\, x \in [a]\}$$

$$a^T = \vee\{x,\, x \in [a]\}$$

[16] Relever un objet consiste à prendre son image réciproque par un morphisme.

L'ensemble des blocs d'une relation de tolérance T sur L peut être ordonné par l'ordre induit sur les bornes inférieures (dualement supérieures). Ainsi ordonné, cet ensemble L/T, est le treillis quotient de L par T .

La notion de relation de tolérance peut être abordée avec un autre point de vue, en utilisant la notion suivante :

Définition 2.3.29. Une relation *bloc* S sur un contexte $K = (O, A, R)$ est une relation $S \subset O \times A$ telle que :

- $R \subseteq S$
- pour tout $o \in O$, o^S est une intension de K.
- pour tout $a \in A$, a^S est une extension de K.

Deux concepts de K (C_1, B_1), (C_2, B_2) sont en relation par S si et seulement si :

$$C_1 \times B_2 \cup C_2 \times B_1 \subseteq S$$

Etant donnée une relation bloc S sur le contexte $K = (O, A, R)$, on en déduit une relation de tolérance T sur le treillis de concept L en posant :

$$(C_1, B_1)\, T\, (C_2, B_2) \Longleftrightarrow C_1 \times B_2 \cup C_2 \times B_1 \subseteq S$$

Ensuite, nous avons le théorème d'équivalence suivant [88] :

Théorème 2.3.8. *L'ensemble des relations blocs sur un contexte est en bijection avec l'ensemble des relations de tolérances sur son treillis de concepts.*

Plus précisément, si $K = (O, A, R)$ est un contexte, L_R son treillis de concepts, S une relation bloc sur K, L_S son treillis de concepts (de (O, A, S)) et T la relation de tolérance correspondante, alors nous avons :

$$L_R/T \simeq L_S$$

et si (C, B) est un concept de L_S, $[(B^R, B), (C, C^R)]$ est le bloc correspondant, de plus $(C, B, R \cap C \times B)$ est un contexte pour ce bloc.

Exemple 2.3.19. Considérons la figure 2.5 basée sur l'exemple des couleurs. Sur la partie gauche est présenté le recouvrement des noeuds défini une relation de tolérance. Trois blocs de cette relation sont représentés ; ce sont des intervalles. Les hauts de ces intervalles sont $Black$, $Maroon$ et $Pink$. Les bas des intervalles sont $LightGreen$, $LightYellow$ et $White$. Sur la partie droite, le treillis quotient correspondant.

Maintenant, étant donné un contexte (ou un treillis L), l'ensemble des relations blocs (ou tolérances sur L) peut être ordonné par l'inclusion. On obtient ainsi le treillis des tolérances $\mathrm{Tol}(L)$ où l'infimum est l'intersection et le supremum est la plus petite tolérance contenant l'union.

Nous avons les résultats suivants concernant ce treillis [35, 142] :

Théorème 2.3.9. *Soit un treillis L et $a, b \in L$ tels que b soit un successeur immédiat de a. Alors il existe une plus petite tolérance T_{ab} telle que $aT_{ab}b$.*

De plus, cet ensemble de tolérances T_{ab} contient l'ensemble des inf-irréductibles de $Tol(L)$.

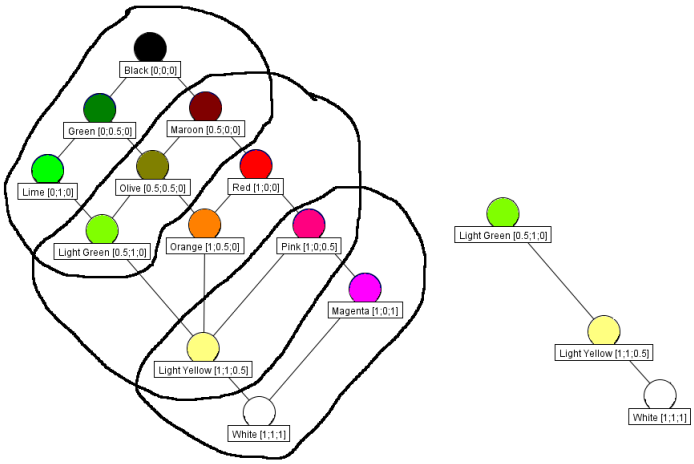


FIGURE 2.5 : Sur la partie gauche, un recouvrement des noeuds défini par une relation de tolérance. Sur la partie droite, le treillis quotient correspondant.

2.3.10 La construction de doublement de convexe

La décomposition présentée dans la section 4.3 est basée sur la construction de doublement d'A. Day [47, 49, 50, 139]. On rappelle qu'une partie $C \subseteq L_C$ d'un treillis L_C est convexe si elle vérifie la condition suivante : pour tout $x, y \in C$ tel que $x \leq y$, si $z \in L_C$ est tel que $x \leq z \leq y$ alors $z \in C$. En résumé, un convexe contient tout ses intervalles.

Nous pouvons maintenant donner la construction de doublement de convexe d'A. Day. Soit $C \subset L_C$ un convexe du treillis L_C, on note $L_C[C] = (L_C \backslash C) \cup (C \times \{0, 1\})$. On définit l'ordre suivant sur $L_C[C]$:

- $x \leq y$ si $x \leq y$ dans L_C
- $(x, i) \leq y$ si $x \leq y$ dans L_C
- $x \leq (y, j)$ si $x \leq y$ dans L_C
- $(x, i) \leq (y, j)$ si $x \leq y$ dans L_C et $i \leq j$.

Ordonné ainsi $L_C[C]$ est un treillis.

Par exemple, considérons le treillis L_C de la figure 2.6.

Le convexe C est identifié par des noeuds en rouge. Après avoir appliqué la construction d'A. Day, on obtient le treillis de la figure 2.7 Les deux occurrences du convexe sont identifiées par des noeuds en vert et rose.

Notre construction est basée sur les deux théorèmes suivants. Le premier est prouvé par W. Geyer [89].

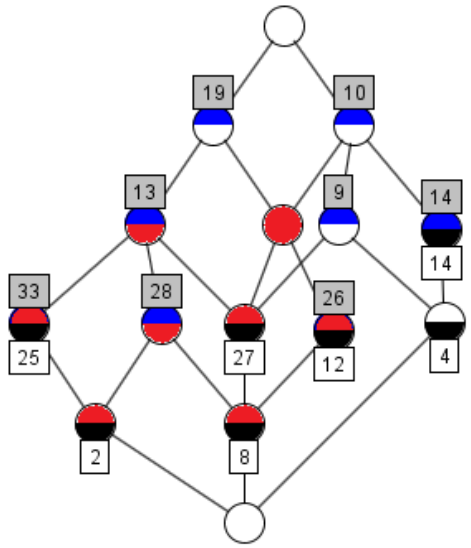


FIGURE 2.6 : Un treillis où les concepts-objets $(\{o\}'', \{o\}')$ ont un nombre au-dessous et les concepts-attributs $(\{a\}', \{a\}'')$ ont un nombre au-dessus.

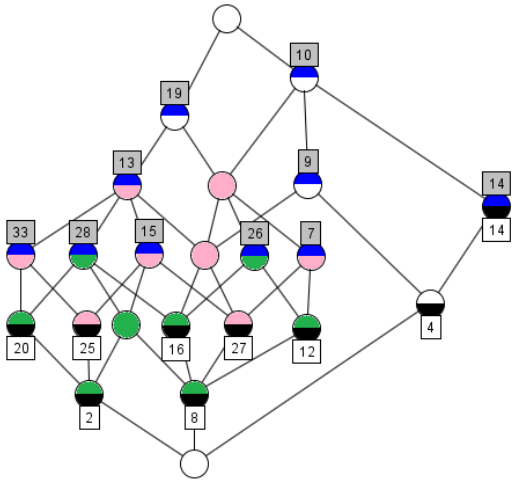


FIGURE 2.7 : Treillis de la figure 2.6 dont le convexe (noeuds en rouge) a été doublé. Les noeuds de chaque convexe sont en rose et vert.

Théorème 2.3.10. *Un treillis $L = L_C[C]$ este obtenu par le doublement du convexe C dans le treillis L_C si et seulement si, en notant $K = (J_L, M_L, R)$ le contexte réduit de L, il existe $J \subseteq J_L$ et $MsubseteqM_L$ tels que $K_C = (J, M, R \cap J_L \times M_L)$ soit le contexte réduit de L_C et soit un sous-contexte flèche-fermé de K tel que :*

$$R \cap ((O \backslash J) \times (A \backslash M)) = \emptyset \tag{2.9}$$

Dans la suite nous ferons référence à cette condition sous l'appellation condition de Geyer. Nous sommes maintenant amenés à chercher des sous-contextes flèche-fermés qui vérifient la condition de Geyer.

Avec le théorème 2.3.6, c'est équivalent de chercher un sous-contexte flèche-fermé et une relation de congruence. Maintenant, on peut également utiliser le fait que l'ensemble des relations de congruence peut être ordonné par l'inclusion et hérite ainsi d'une structure de treillis. De plus, l'intersection de deux relations de congruence est une relation de congruence ; on obtient ainsi l'opération d'infimum. L'opération de supremum est comme d'habitude moins simple. Cependant, étant donné un ensemble de relations de congruence, il suffit de considérer la plus petite relation de congruence qui les contient toutes.

Avec cette structure de treillis sur l'ensemble des relations de congruence, on peut énoncer le théorème suivant d'A. Day [49] :

Théorème 2.3.11. *Soit L un treillis normalement congruent et Θ une relation de congruence qui est un atome*[17] *du treillis des congruences. Alors, il existe un convexe $C \subseteq L$ tel que L soit isomorphe à $(L/\Theta)[C]$.*

La notion de normalement congruent n'a pas été définie car elle ne sera pas utilisée. Remarquons également que la preuve d'A. Day est plus explicite concernant le convexe C, mais nous utiliserons une nouvelle construction.

2.4 Autour de la logique

Il y a de nombreuses logiques, par exemple classique (Aristote), intuitionniste, floue, etc. Un des points de rencontre entre la FCA et la logique est la recherche d'implications lesquelles sont liées aux dépendances fonctionnelles dans les bases de données [11].

Remarque : Dans cette section, nous utiliserons une autre notation pour les contextes, à savoir $K = (G, M, I)$ au lieu de $K = (O, A, R)$ car cette section traite des implications qui sont usuellement notées $A \to B$.

2.4.1 Bases d'implications

Parmi les directions de recherche étudiées par la FCA, la recherche d'implications est probablement celle qui a suscité le plus d'intérêts. En effet, ces implications sont liées d'une part aux dépendances fonctionnelles dans les bases

[17] Un atome est un successeur de l'élément minimal.

de données et d'autre part aux règles d'association dans le domaine de la fouille de données.

Les définitions formelles sont les suivantes :

Définition 2.4.1. Etant donné un contexte $K = (G, M, I)$, une implication (entre attributs) $A \rightarrow B$, avec $A, B \subseteq M$, est valide dans K si et seulement si toute observation possédant tous les attributs de A, possède aussi tous les attributs de B (i.e. $A' \subseteq B'$). L'ensemble A est alors appelé la prémisse et B la conclusion.
L'ensemble de toutes les implications valides d'un contexte K est noté Γ, et si nécessaire Γ_K.

Lorsque $B \subseteq A$, l'implication $A \rightarrow B$ est valide mais ne présente pas d'intérêt.

Définition 2.4.2. Une implication valide $A \rightarrow B$ est dite informative lorsque $B \not\subseteq A$.

Remarques :

- Si A n'est pas fermé ($A \neq A''$), alors $A \rightarrow A''$ est informative. Ainsi lorsque l'on recherche des implications, on se concentre en général sur les prémisses. La conclusion implicite est alors la fermeture de la prémisse.
- Une implication est un cas particulier de règle d'association puisque sa confiance vaut un.

L'ensemble Γ de toutes les implications est en général très grand, et contient des implications non-informatives ; il est donc d'un intérêt essentiellement théorique, comme ensemble accessible.
En effet, il est possible de déduire des implications à partir d'autres en se servant d'ensembles d'axiomes. Parmi ces ensembles, les plus couramment utilisés sont ceux d'Armstrong [9] :

- Axiome de réflexivité : si $B \subseteq A$ alors $A \rightarrow B$.
- Axiome d'augmentation : si $A \rightarrow B$ alors $A \cup C \rightarrow B \cup C$
- Axiome de transitivité : si $A \rightarrow B$ et $B \rightarrow C$ alors $A \rightarrow C$

Un ensemble alternatif d'axiomes, appelé logique de simplification pour les Dépendances Fonctionnelles, a été développé [38, 136, 137, 160] :

- Axiome de réflexivité : si $B \subseteq A$ alors $A \rightarrow B$.
- Axiome de fragmentation : si $A \rightarrow B \cup C$ alors $A \rightarrow B$.
- Axiome de composition : si $A \rightarrow B$ et $C \rightarrow D$ alors $A \cup C \rightarrow B \cup D$.
- Axiome de simplification : si $A \subseteq C$, $A \cap B = \emptyset$,alors les deux implications $A \rightarrow B$ et $C \rightarrow D$ peuvent être remplacées par $A \rightarrow B$ et $C \backslash B \rightarrow D \backslash B$.

Remarques :

- Cet ensemble amélioré d'axiomes a permis de mettre en oeuvre de nombreux algorithmes efficaces [159, 160].
- Les auteurs de ces axiomes ont également prouvé qu'ils étaient logiquement équivalents à ceux d'Armstrong, en effet ces deux ensembles d'axiomes permettent de déduire toutes les implications valides.

Dans ce cadre, le challenge est alors de trouver de bons ensembles d'implications qui permettent de déduire toutes les autres.

Définition 2.4.3. Soit un contexte $K = (G, M, I)$ et son ensemble d'implications Γ. Un sous-ensemble $B \subseteq \Gamma$ est une base d'implications si toute implication de Γ peut être déduite de B par l'un ou l'autre des systèmes d'axiomes et si aucune implication de B ne peut être déduite des autres (non-redondance).

Trois différentes bases ont été très largement étudiées :

- La base canonique, également nommée base de Guigues-Duquenne ou *stem basis* [97, 144, 86, 170, 171]
- La base canonique directe qui a été découverte plusieurs fois sous de multiples appellations [25, 162], en particulier base générique.
- La base canonique directe ordonnée [3, 2].

Plus généralement, dans le domaine de la fouille de données, on considère les règles d'association [173] dont la confiance peut être inférieure à 1. Ces règles d'association sont principalement calculées avec l'algorithme Apriori ou l'une de ses multiples améliorations [100].

La FCA s'intéresse également aux implications faisant intervenir des attributs négatifs [134, 157, 158] ; ces nouvelles implications apportent un savoir supplémentaire extrait des données.

Dans la suite, nous aurons besoin de quelques détails supplémentaires concernant à la fois la base canonique et la base canonique directe. Comme nous l'avons déjà mentionné, nous nous focaliserons sur les prémisses :

Définition 2.4.4.

- Les prémisses de la base canonique (Guiges-Duquenne) sont appelées les pseudo-intents.
- Les prémisses de la base canonique directe sont appelées les générateurs minimaux.

Cette définition n'est pas la définition habituelle. En général, la prochaine définition est donnée, puis il est prouvé que les deux sont équivalentes.

Définition 2.4.5. Soit $K = (G, M, I)$ un contexte.

- Etant donné un ensemble fermé d'attributs $A = A''$, une partie $B \subseteq M$ est un générateur de A lorsque $B'' = A$.
- Parmi tous les générateurs d'un ensemble fermé, certains sont minimaux pour l'inclusion ; ce sont les générateurs minimaux.
- Une partie $A \subset M$ est dite quasi-fermée lorsqu'elle n'est pas fermée et qu'elle vérifie la propriété suivante :

$$\forall B \subsetneq A, B'' \subset A$$

- Parmi tous les quasi-fermés, les pseudo-intents sont les minimaux.

2.4.2 Relations entre sup-irréductibles

Des relations ont été introduites permettant d'apporter un autre point de vue sur les bases d'implications [135, 25]. Commençons avec le cas de la base canonique directe. Cette base peut être vue comme la base associée à la relation δ au sens suivant :

Définition 2.4.6. Soit φ un opérateur de fermeture sur un ensemble fini S. La relation de dépendance δ est définie sur les parties de S par :

$$\forall x, y \in S,\ x \delta_X y \iff \exists X \subseteq S,\ x \notin \varphi(X),\ y \notin \varphi(X),\ x \in \varphi(X \cup y)$$

Remarques :

- La relation δ peut également être définie en termes de sup-irréductibles dans un treillis [22] :

$$j \delta_x j' \iff j \not\leq x,\ j' \not\leq x, j < j' \vee x$$

- Enfin, cette relation δ peut aussi être définie à l'aide de flèches [135] :

$$j \delta j' \iff \exists m,\ j \uparrow m,\ j' \not\leq m$$

Alors, la base canonique directe est l'ensemble des implications [25] :

$$\Sigma_\delta = \{X \cup y \to x,\ x \delta_X y \text{ et } X \text{ minimal}\}$$

De même, la base ordonnée directe [3, 2, 125] est la D-base de la relation D [34] :

$$j D j' \iff j = j' \text{ or } \exists x,\ j < j' \vee x,\ j \not\leq j'^{-} \vee x$$

Ou également en termes de flèches [135] :

$$jDj' \Longleftrightarrow \exists m,\, j \uparrow m,\, j' \downarrow m$$

Donnons une nouvelle équivalence, complétant celles données précédemment [93] :

Théorème 2.4.1. *Etant donné un contexte, l'ensemble de ses sous-contextes flèche-fermés correspond bijectivement à celui des D-fermés* [18].

Ce théorème sera utilisé pour relever des implications à partir de treillis quotients dans la section 4.2.

2.5 Aspects algorithmiques

Au cours des trente dernières années, la communauté FCA a développé un grand nombre d'algorithmes très efficaces pour le calcul des treillis de concepts à partir d'un contexte. On peut distinguer quelques propriétés importantes de ces algorithmes :

- Bien sûr, il y a en premier la complexité. L'algorithme connu, ayant la meilleure complexité dans le pire des cas, est celui de L. Nourine et O. Raynaud [143]. Cependant, dans la pratique, ce n'est pas le plus efficace [118, 119].
- Certains algorithmes ne calculent que les concepts, i.e. les fermés, d'autres calculent entièrement le diagramme de Hasse du treillis [27, 86, 96].
- Certains algorithmes ont été spécialement développés pour le calcul parallèle [81, 82, 109, 111] ou le calcul distribué [13, 122], ou bien sont des adaptations parallélisées/distribuées des algorithmes classique [110] ; ceci dans le but d'obtenir une bonne capacité d'expansion (*scalability*). D'autres techniques de passage à l'échelle ont également été étudiées, en particulier sur des données médicales [37].
- Certains algorithmes ont besoin d'avoir accès à l'ensemble du jeu de données pour calculer le treillis de concepts, d'autres permettent un traitement "online" de données. Il s'agit des algorithmes incrémentaux [90, 172, 174, 99].

Pour conclure, l'algorithme le plus célèbre pour le calcul d'un treillis de concepts est celui de B. Ganter [86] (NextClosure) ; en pratique, c'est le plus efficace et donc certainement le plus utilisé.

L'algorithme le plus connu pour le calcul de la base de Guigues-Duquenne est NextClosure [86] et sa version parallèle [113]. Un algorithme basé sur les clauses de Horn a également été développé [8].

[18] Un sous-contexte est D-fermé si lorsqu'il contient un attribut alors il contient tous ceux avec qui il est en relation D.

2.6 Aspects visualisations

Une visualisation très simple, utilisée dans [18], peut être obtenue en montrant uniquement un concept, ses successeurs immédiats et ses prédécesseurs immédiats.

Dans la suite, nous utiliserons des relations de congruence pour décomposer des treillis. Le treillis quotient déduit d'une relation de congruence peut également être utilisé afin de construire une visualisation plus simple d'un treillis donné [59, 60].

En lien avec ces aspects de visualisation, il est également important de mentionner la problématique de la navigation. En effet, les treillis sont particulièrement adaptés à la navigation dans les données. Très basiquement, il est possible de se déplacer via la relation parent/enfant entre les concepts. Bien sûr, des approches plus élaborées ont déjà été implémentées [31, 177], en particulier la navigation par facettes [70, 150, 151, 152]. Notre projet "LattExp" (chapitre 3 et section 6.1) propose une nouvelle méthode de navigation.

Ensuite, il y a des visualisations plus élaborées qui permettent de représenter le diagramme de Hasse du treillis (ou "line diagram"), ou même les plus élaborés "nested line diagram" (diagramme en ligne imbriquées).

La représentation la plus répandue est basée sur le standard allemand DIN 2331 dont les règles sont :

- Chaque noeud est un concept
- Un concept est au-dessus d'un autre si et seulement si c'est un super-concept [19] de celui-ci.
- Pour tout $o \in O$, o est écrit sous le concept $(\{o\}'', \{o\}')$.
- Dualement, a est écrit au-dessus du concept $(\{a\}', \{a\}'')$, pour tout $a \in A$.
- En conséquence, l'extension d'un concept consiste en l'ensemble des o au-dessous de ce concept et son intension est l'ensemble des a au-dessus.

Pour mettre en oeuvre des algorithmes de représentation respectant ce standard, une autre représentation a été définie connue sous le nom de méthode géométrique [167, 185]. Cette méthode géométrique est utilisée comme étape intermédiaire avant la construction du "line diagram". Cependant, elle peut également être utilisée directement comme une visualisation alternative d'un treillis. Une implémentation récente a été faite par E. Bertret, F. Bretaud, B. Feix et sera bientôt disponible dans le projet "TheGalactic".

D'autres algorithmes de représentations existent, en particulier un développé par R. Freese [79] basé sur une notion de force répulsive entre les noeuds et un autre basé sur un algorithme génétique [147] qui donnent tous deux d'excellents résultats.

Nous conclurons ces paragraphes en rapport avec la visualisation par deux remarques :

[19] Concept supérieur pour la relation d'ordre.

- Une étude [66] de P. Eklund and J. Villerd sur les visualisations a déjà été publiée et donne un point de vue plus général au sujet de la representation des données.

- D'une manière générale, les algorithmes de représentation de treillis donnent de beaux résultats pour des treillis ayant jusqu'à $40 - 50$ noeuds. Cependant, le logiciel GLAD était capable de représenter de beaux treillis jusqu'à une centaine de noeuds. Nous regrettons sincèrement que ce logiciel ne soit plus disponible.

2.7 Généralisations de la FCA

A partir du cadre initial de la FCA, de nombreuses généralisations ont été définies. Avec la FCA classique, un contexte binaire est donné et peut être interprété par la phrase suivante "une observation possède un attribut".

Nous proposons la classification suivante des théories généralisant la FCA, ceci en généralisant la phrase précédente :

- "une observation possède partiellement un attribut" est la phrase correspondant à l'analyse de concepts flous. Dans le même esprit, l'analyse triadique des concepts [105] correspond à la phrase "une observation possède une propriété sous une condition".

- Il y a un besoin de travailler avec des données complexes, par exemple des graphes pouvant représenter des molécules. Certains algorithmes sont définis spécifiquement, pour un problème donné [191] ; mais plus généralement, il existe des cadres théoriques permettant d'aborder de manière globale ces données complexes. On peut citer en particulier : l'analyse des concepts logiques, l'analyse de données symbolique et les structures de motifs ("*pattern structures*") [87]. Ces approches correspondent toutes à la phrase "une observation possède une description complexe".

- Dans beaucoup de situations pratiques, il s'avère insuffisant de ne considérer qu'un seul contexte. Pour cela, plusieurs cadres permettant de travailler avec plusieurs contextes ont été définis [183, 187]. En particulier, il est possible de généraliser l'approche des bases de données relationnelles et de lier plusieurs contextes entre eux. Cette problématique est le champ d'investigation de l'analyse relationnelle des concepts (RCA) [102, 161].

2.7.1 L'approche floue (« fuzzy »)

Plusieurs généralisations floues de la FCA ont été envisagées ; succintement, le point de vue "un objet possède un attribut" est remplacé par "un objet possède partiellement un attribut". Rigoureusement, les valeurs de vérité $\{0, 1\}$ sont remplacées par un treillis résidué L ou même un multi-treillis (voir par exemple [128] pour les définitions formelles). Les théorèmes fondamentaux de la FCA ont été généralisés au cadre flou [17]. Avec un tel cadre, il est possible de travailler

avec des données bruitées via des concepts flous. Il est également possible de considérer des implications dont la confiance n'est pas 100% à l'aide de la logique floue.

Remarque : Notre projet LattExp (chapitre 3) utilise le treillis des relations de tolérances comme espace de navigation. Ce treillis est un treillis résidué [16], de sorte qu'il peut également être utilisé comme un outil intrinsèque de "fuzzification".

2.7.2 L'approche par « patterns »

Parmi toutes les généralisations de la FCA, la direction la plus active actuellement, en 2015, est celle utilisant les "pattern structures" (structures de motifs) [87]. Cette généralisation de la FCA permet de travailler avec des données complexes [114] (Annexe 7.5.5) mais également avec de simples données numériques [106, 107]. Avec ce point de vue, il est également possible de calculer les dépendances fonctionnelles [12]. Des tentatives antérieures ayant pour objectif de gérer des données complexes avaient eu lieu [131, 130, 124] mais avec moins de succès.

Une recherche de motifs generalisés [121] a également été mise en place et constitue une autre approche dans la fouille de données, liée à la FCA et à l'usage de taxonomies. Cette approche permet la découverte de nouvelles connaissances à partir des données.

2.7.3 L'analyse de données symboliques

De nombreux travaux sur l'analyse de données symboliques (SDA) existent [149, 52, 53, 29]. Cette théorie permet de traiter de données complexes : histogrammes, probabilités, etc. Les liens entre cette approche et celle par patterns sont évidemment très forts [4, 108].

Comme dans le cadre des pattern structures, chaque objet/observation possède une description, sous forme d'un objet symbolique (objet formé par exemple d'un histogramme, d'un intervalle, d'une valeur numérique, etc.). Cet espace de description est muni d'une structure de treillis qui permet alors d'en déduire un treillis de concepts.

2.7.4 Analyse de concepts logiques

L'analyse de concepts logiques (LCA) est très proche de l'approche par patterns. En effet, avec la LCA, les observations ne sont plus décrites par des patterns, mais par des formules logiques [71, 68, 72, 69, 70].

Un cadre abstrait a été ajouté à cette approche, la navigation conceptuelle abstraite (ACN) qui sera décrite avec un peu plus de détails à la section 2.9

2.7.5 Analyse de concepts relationnels

L'analyse des concepts relationels [161] créée un pont entre les bases de données relationelles et la FCA. En effet, cette généralisation de la FCA permet de considérer plusieurs contextes liés les uns aux autres. Les concepts déduits de ces contextes ne sont plus déduits à partir d'un seul contexte mais de plusieurs ; ils sont donc d'une nature plus riche.

2.8 Applications de la FCA

A ses tous débuts, la FCA était essentiellement une théorie mathématique. Cependant, ses problématiques principales sont rapidement devenues de nature informatique [165].

Les treillis de concepts apportent une structure mathématique particulièrement bien adaptée à la classification et l'ordonnancement des données. Ainsi, en parallèle du développement de l'Analyse de Concepts Formels, un très grand nombre d'applications dans une très grande variété de domaines ont été mises en oeuvre. La liste suivante ne vise pas à l'exhausitivité mais souhaite montrer l'extrême diversité de ces applications :

- Sciences Sociales [63, 62], par exemple pour l'analyse de questionnaires [15, 18], l'analyse de handicaps psychologiques [57], la définition de protocoles experimentaux [56]. En relation avec ces applications, l'analyse des réseaux sociaux est une préoccupation de longue date de la communauté FCA [75, 73, 74, 163, 117].
- Fouille dans les données textuelles [85, 152], en particulier les systèmes CREDO [32], et Mail-Sleuth [65].
- Fouille dans les données médicales [40], en particulier l'expression des gènes [103, 2]. En lien avec ces applications, celles liées à la chimie [120], en particulier la fouille dans les protéines ou les enzymes [39], doivent apparaître.
- Extraction de connaissances [55].

Deux conférences internationales sont spécialisées dans la FCA : *Concept Lattices and their Applications* (CLA [101]) et *International Conference on Formal Concept Analysis* (ICFCA [23]). Les dernières occurences de ces conférences (resp. 2016 et 2017) ont permis la diffusion d'applications récentes de la FCA dans des directions extrêment variées.

2.9 RI, EC et FCA

Nous avons brièvement présenté les problématiques de la recherche d'informations et de l'extraction de connaissances. Ensuite, nous avons présenté de manière détaillée l'analyse de concepts formels. Cette section est dédiée à une

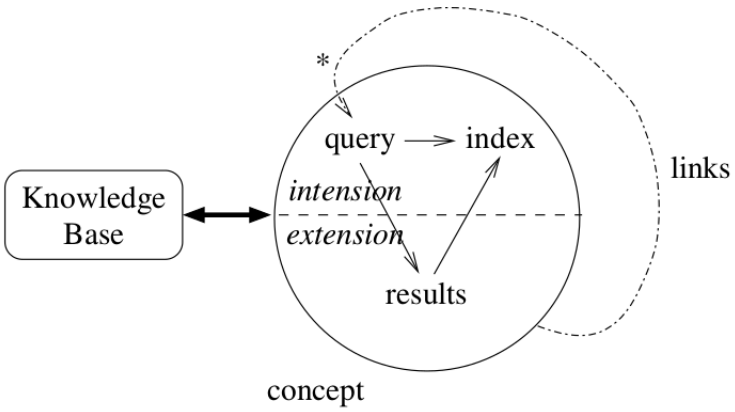


FIGURE 2.8 : Elements et fonctionnement d'une instance d'ACN

brève présentation de l'ACN (Navigation Conceptuelle Abstraite) qui utilise la FCA pour de la recherche d'informations et de l'extraction de connaissances. Bien que notre projet LattExp soit moins développé, il est dans le même esprit, tout en utilisant une approche et des outils différents. D'autres projets de ce type existent, par exemple Mail-Sleuth [65], FaIR [152], CREDO [32], mais nous avons choisis l'ACN qui est un cadre à la fois récent et riche en terme d'expressivité.

L'ACN est "concept-centrée"[20] : les concepts ont le rôle fondamental défini à l'origine de la FCA d'"Unité de Base de la Pensée" (figure 2.8 [70]). Il est également important de noter que l'ACN est Abstraite, au sens de la programmation ; elle doit donc être instanciée. Ainsi de nombreux cadres existant peuvent être considérés comme des instances de l'ACN.

Rappelons que les composants principaux d'un cadre complet pour la recherche d'informations et l'extraction de connaissances sont : un langage de requêtes, un outil de navigation, et des vues sur les données.

Une instance de l'ACN vient donc avec :

- Une base de connaissances contenant les données prétraitées.
- Un langage de requêtes dont l'expressivité dépend de l'instance considérée et de la base de connaissances. Une requête correspond à la fois à l'intention de l'utilisateur et à l'intension du concept.
- Une extension qui, à nouveau, correspond à la fois au résultat de la requête et l'extension du concept.
- Un index fait de caractéristiques sur les données, utilisé pour aider l'utilisateur dans sa navigation.

[20] Le "C" est au centre de "ACN"

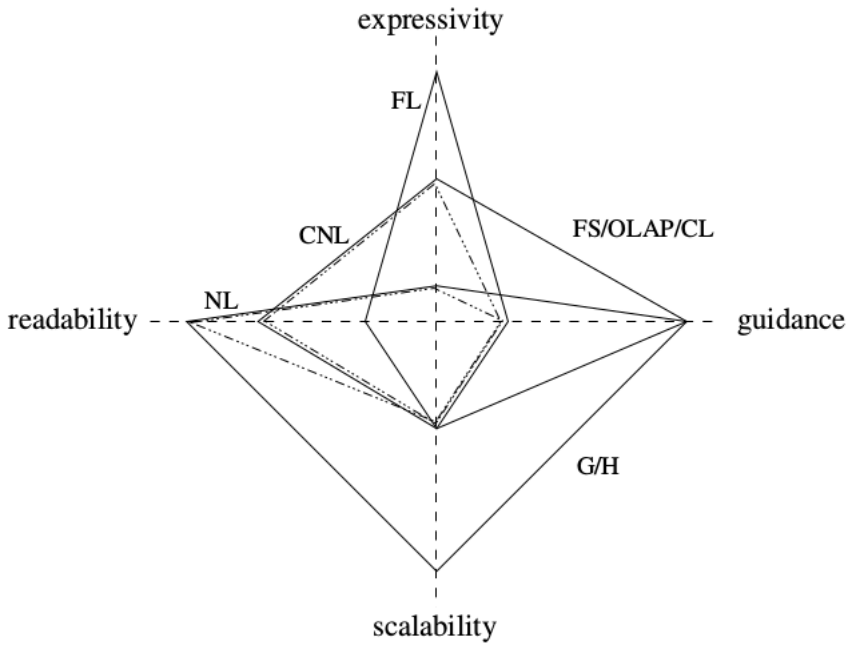


FIGURE 2.9 : Caractéristiques principales qu'une instance d'ACN peut avoir (NL : Natural Langage, CNL : Controlled NL, FL : Formal Langage, FS : Faceted Search, CL : Concept Lattices, G/H : Graphs, Hierarchies) et les outils de référence. (Source du schéma [70])

- Des liens utilisés pour la navigation effective à travers les concepts.

Chaque instance de l'ACN essaie de maximiser sa région dans la figure 2.9.

Notre projet, LattExp, est encore jeune, mais il possède déjà toutes les graines pour devenir une bonne instance de l'ACN.

Deuxième partie

Contributions

Chapitre 3

LattExp : a Lattice Explorer

Ce chapitre est dédié à la présentation de notre proposition, nommée LattExp. Ce cadre est basé sur l'Analyse de Concepts Formels (FCA). Lors d'une première tentative, nous avons essayé de concilier la FCA et les données massives. Cependant, d'après beaucoup de membres de la communauté FCA, cela ne semble pas possible, au moins pour les raisons suivantes :

- La FCA a besoin de données déjà un peu structurées, alors que le paradigme des données massives fournit une grande variété de types de données.
- La FCA fournit des treillis de concepts et des bases d'implication dont la taille peut être exponentielle en la taille des données initiales. Les données massives fournissent des données initiales qui sont déjà en très grand nombre.
- La visualisation des treillis de concepts est une force de la FCA. Cependant, les approches classiques permettent de visualiser autour de 40-50 concepts et GLAD était capable de fournir de belles visualisations jusqu'à une centaine (figure 3). Cela n'est bien sûr plus possible dans le cadre des données massives.

Nous avons produit des algorithmes distribués utilisant Hadoop pour faire des calculs pour la FCA. Ils sont donnés au chapitre 5, mais ne font pas partie du projet LattExp.

Rappelons quelques points que nous avons mentionnés dans les chapitres précédents :

- La recherche d'informations (RI) fournit des méthodes permettant d'ordonner puis sélectionner les données à travers l'usage d'un langage de requêtes.
- L'extraction de connaissances (EC) fournit des outils de découverte de motifs de divers types (règles d'association, implications, groupes, règles de classification, exceptions).

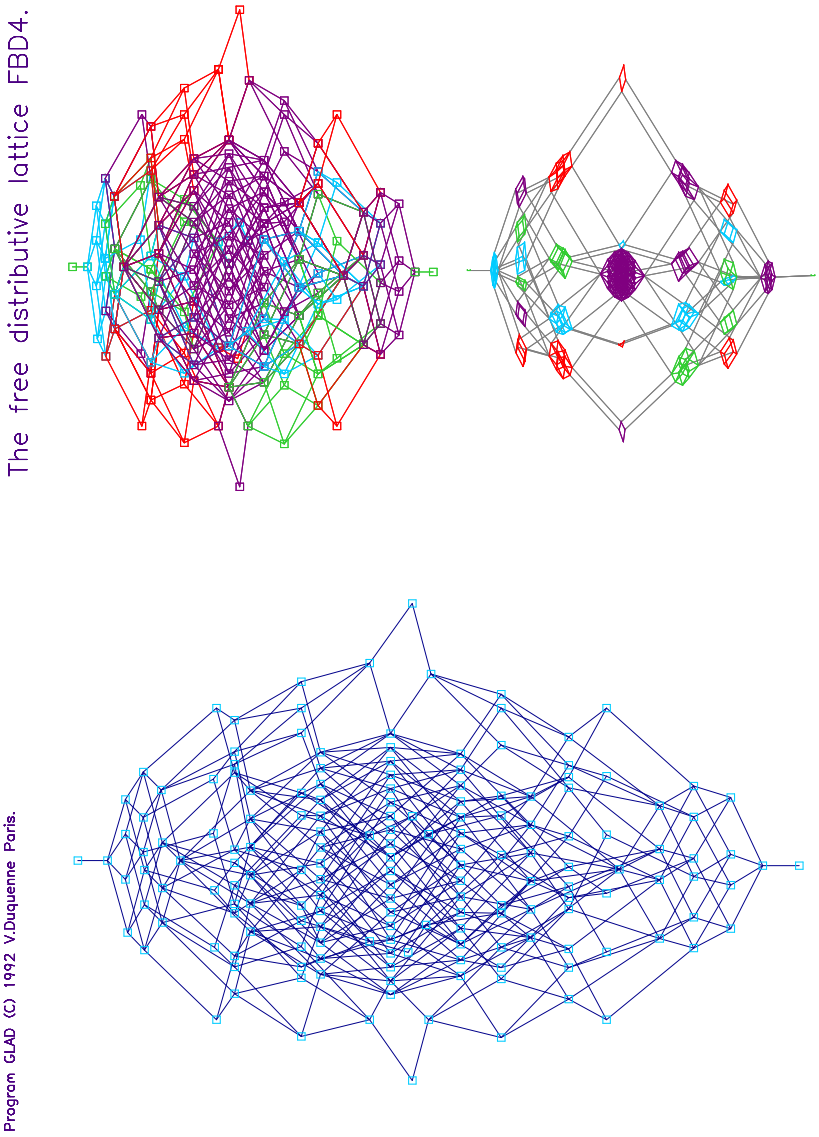


FIGURE 3.1 : Une visualisation produite par GLAD.

- Complémentaire à la RI et l'EC, la FCA fournit les treillis de concepts qui forment une structure navigable à la fois classifiante et ordonnante.

Décrivons maintenant notre projet LattExp :

- Nous avons présenté l'ACN comme étant une analyse "centrée-concept". LattExp peut lui être présenté comme "centré-treillis". Dans un certain sens, il y a un glissement du local (concept) au global (treillis). Cela signifie que le concept n'est plus seulement "l'unité de base de la pensée" mais qu'il est aussi une partie dans un environnement, liée à d'autres concepts.
- LattExp se concentre sur les aspects navigation et visualisation, ce qui signifie qu'il propose à l'utilisateur de naviguer à travers des visualisations de treillis.
- Cette navigation s'effectue en utilisant le treillis des relations de tolérances qui joue le rôle de carte du monde. Il n'est cependant par indispensable de le calculer pour naviguer.
- LattExp fournit également des métriques (valeurs calculées à partir de l'étape courante, vers les étapes suivantes potentielles) afin de guider l'utilisateur dans sa navigation.

Finalement, LattExp n'est pas si loin d'être une instance d'ACN :

- Il a une base de connaissances factuelles (i.e. une base de faits), qui est le contexte binaire d'entrée.
- Il a un opérateur d'extension dont les valeurs sont les différents treillis quotient que l'on peut calculer.
- Les métriques, en tant que guide pour la navigation, jouent le rôle d'index.
- Chaque treillis quotient est lié à ses prédécesseurs et successeurs pour les prochaines étapes de navigation.
- Le "seul" (et non des moindres) point manquant est le langage de requête, mais cela sera discuté au dernier chapitre.

Un cas d'usage de LattExp est donné à la section 6.1.

Chapitre 4

Décompositions de treillis

Sommaire

4.1 Décomposition sous-directe

Commençons avec les definitions :

Définition 4.1.1.

- Soit L un treillis. Un sous-treillis de L est une partie $S \subseteq L$ de L stable par les opération inf et sup et qui est un treillis pour ces opérations induites.
- Etant donnés L_i, $i \in \{1, \ldots, n\}$ des treillis, le produit cartésien $L_1 \times \cdots \times L_n$ est naturellement muni d'une structure de treillis ; ce treillis $L_1 \times \cdots \times L_n$ est nommé produit direct des L_i.

Définition 4.1.2. Un produit sous-direct est un sous-treillis d'un treillis produit (direct) $L_1 \times \cdots \times L_n$ de treillis $L_i, i \in \{1, \ldots, n\}$ tel que les projections

	b	c	d	f	g	k
2	×	×	×	×	×	↕
3	×	↕	×	↓	×	×
5	×	×	×	↕	↕	○
6	×	×	↕	×	↓	○
9	↕	×	○	×	○	○

TABLE 4.1 : Contexte réduit du treillis de la table 2.3 complété avec les flèches.

$\pi_i : L \to L_i$ sur les facteurs soient des morphismes surjectifs. Les treillis $L_i, i \in \{1, \ldots, n\}$ sont appelés treillis facteurs et sont des treillis quotients. Une décomposition sous-directe d'un treillis L est un isomorphisme entre L et un produit sous-direct, ce que l'on peut noter :

$$L \hookrightarrow L_1 \times \cdots \times L_n \twoheadrightarrow L_i$$

Exemple 4.1.1. Nous illustrerons notre approche de la décomposition sous-directe à l'aide de la table 4.1.

4.1.1 Résultat principal

A partir des équivalences rappelées précédemment (Théorèmes 2.3.4 et 2.3.6), présentes dans [88], on peut déduire les corollaires suivants :

Corollaire . *Etant donné un treillis L et son contexte réduit (O, A, R), sont en bijection :*

1. *L'ensemble des sous-contextes flèche-fermés de (O, A, R),*
2. *L'ensemble des sous-contextes compatibles de (O, A, R),*
3. *L'ensemble des relations de congruence de L et leur treillis quotient.*

Définition 4.1.3. Etant donné un contexte (O, A, R), une famille $(O_i, A_i, R \cap O_i \times A_i)_{i \in I}$ convre ce contexte lorsque :

$$\cup_{i \in I} O_i = O, \cup_{i \in I} A_i = A$$

Corollaire . *Etant donné un treillis L et son contexte réduit (O, A, R), sont en bijection :*

1. *Les familles de sous-contextes flèche-fermés de (O, A, R) couvrant O et A,*
2. *Les familles de sous-contextes compatibles de (O, A, R) couvrant O et A,*

3. *Les familles $(\theta_i)_{i\in I}$ de relations de congruence de L telles que $\cap_{i\in I}\theta_i = \Delta$ où $x\Delta y \Longleftrightarrow x = y$.*

4. *L'ensemble des décompositions sous-directes de L et leurs treillis facteurs.*

Dans la suite, nous exploitons ces quatres points de vue.

1. La décomposition sous-directe assure que L est un sous-treillis d'un treillis produit-direct. De plus, chaque projection sur un des facteurs est surjective.

2. Les relations de congruences de L nous permettent de voir que les treillis facteurs de la décomposition sont des treillis quotients et donc ils préservent les partitions via les classes d'équivalence.

3. Les sous-contextes compatibles permettent de calculer le morphisme d'injection de L dans le treillis produit-direct.

4. Les sous-contextes flèche-fermés permettent de calculer les contextes réduits des treillis facteurs/quotients.

Dans ce qui suit, nous décrivons la construction d'une décomposition sous-directe particulière.

4.1.2 Calcul des facteurs irréductibles

Dans cette sous-section, nous considérons les décompositions sous-directes d'un treillis L avec en entrée son contexte réduit (O, A, R). Avec les corollaires de la sous-section 4.1.1, une décomposition sous-directe d'un treillis L peut être obtenue en calculant un ensemble de sous-contextes flèche-fermés de (O, A, R) qui couvrent O et A. Il y a évidemment plusieurs recouvrements satisfaisant ces conditions et donc plusieurs décompositions sous-directes. En particulier, la décompostion d'un treillis L en L lui-même, qui correspond à prendre un seul sous-contexte flèche-fermé : le contexte (O, A, R) en entier. Un algorithme de décomposition sous-directe a déjà été proposé [83]. Cependant, toutes les relations de congruence sont calculées et ensuite seulement des paires couvrantes de relations sont considérées. En conséquence, plusieurs décompositions peuvent être obtenues, et ces décompositions possèdent nécessairement seulement deux facteurs.

Ici, nous nous concentrons sur une décomposition sous-directe d'un contexte en un nombre éventuellement grand de facteurs de taille plus petite. Ces derniers étant irréductibles au sens suivant : un treillis L est irréductible lorsqu'il apparaît comme facteur de toutes ses décompositions sous-directes. Une caractérisation des décompositions sous-directes en facteurs irréductibles se trouve dans [88] :

Un contexte (O, A, R) est dit monogène lorsqu'il peut être obtenu par flèche-fermeture d'un contexte contenant un seul $j \in O$. Ainsi (O, A, R) est le plus petit contexte flèche-fermé contenant j.

Proposition 4.1.1. *Un treillis L est irréductible si et seulement si son contexte réduit est monogène.*

Nous pouvons donc déduire le résultat suivant :

Proposition 4.1.2. *Soit L un treillis. On peut déduire de L un treillis produit direct $L_1 \times \cdots \times L_n$ tel que chaque treillis L_i est :*

- *le treillis de concepts d'un sous-contexte monogène ;*
- *irréductible et ;*
- *un treillis facteur d'une décomposition sous-directe.*

Ce résultat peut être mis en oeuvre avec l'algorithme 1, polynomial en temps, qui permet de trouver les contextes des facteurs $L_1, \ldots, L_n$ d'une décomposition sous-directe, avec un contexte (O, A, R) en entrée. En supposant les treillis construits et si le nombre n_J de sup-irréductibles est plus grand que le nombre n_M d'inf-irréductibles, alors la complexité est en $\mathcal{O}(n_J n_M)$. Les sous-contextes monogènes de la forme (J_j, M_j, R_j) sont obtenus par fermeture de chaque $j \in O$, via l'algorithme 2. La décomposition sous-directe de L est alors obtenue en formant les treillis de concepts de ces sous-contextes.

On peut remarquer que les fermetures sont calculées sur les sup-irréductibles,

j	Flèche_Fermeture			$\mathcal{L}$
	Entrée	Sortie		
	$(\tilde{J}, \tilde{M}, \tilde{R})$	J_j	M_j	
2	$(2, \emptyset, \emptyset)$	$\{2\}$	$\{k\}$	$\times$
3	$(3, \emptyset, \emptyset)$	$\{3, 5, 6\}$	$\{c, d, f, g\}$	$\times$
5	$(5, \emptyset, \emptyset)$	$\{5, 6\}$	$\{d, f, g\}$	
6	$(6, \emptyset, \emptyset)$	$\{6\}$	$\{d\}$	
9	$(9, \emptyset, \emptyset)$	$\{9\}$	$\{b\}$	$\times$

FIGURE 4.1 : Itérations de l'algorithme 1 pour le contexte réduit de la table 4.1

mais cela aurait pu être fait sur les inf-irréductibles.

Algorithme 1 : Décomposition Sous-Directe

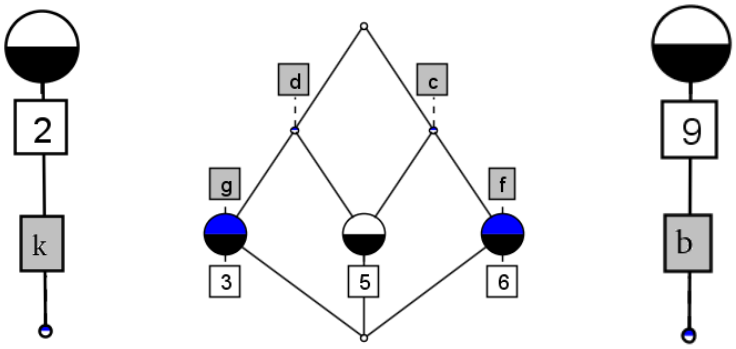
Entrées : Un contexte (O, A, R)
Sorties : Liste $\mathcal{L}$ des sous-contextes (J_j, M_j, R_j) des facteurs irréductibles.

$\mathcal{L} \leftarrow \emptyset$;
pour tous les $j \in O$ **faire**
  Calculer $(J_j, M_j, R_j) =$ **Flèche_Fermeture**$((j, \emptyset, \emptyset), (O, A, R))$, sous-contexte monogène engendré par j;
  si $\mathcal{L}$ *ne contient pas de sous-contexte couvrant* (J_j, M_j, R_j) **alors**
    ajouter (J_j, M_j, R_j) à $\mathcal{L}$
  si $\mathcal{L}$ *contient un sous-contexte* (J, M, R) *couvert par* (J_j, M_j, R_j) **alors**
    effacer (J, M, R) de $\mathcal{L}$
  retourner $\mathcal{L}$;

Algorithme 2 : Flèche Fermeture

Entrées : Un sous-contexte $(\tilde{J}, \tilde{M}, \tilde{R})$ d'un contexte (O, A, R)
Sorties : La flèche-fermeture de $(\tilde{J}, \tilde{M}, \tilde{R})$

$J_j = \tilde{J}$; $M_j = \tilde{M}$;
$pred_J = 0$; $pred_M = 0$;
tant que $pred_M < cardinal(M_j)$ *ou* $pred_J < cardinal(J_j)$ **faire**
  $pred_J =$ cardinal(J_j);
  $pred_M =$ cardinal(M_j);
  pour tous les $s \in J_j$ **faire**
    ajouter à M_j tous les $m \in A$ tels que $s \uparrow m$ ou $s \updownarrow m$;
  pour tous les $m \in M_j$ **faire**
    ajouter à J_j tous les $s \in O$ tels que $s \downarrow m$ ou $s \updownarrow m$;
  Retourner $(J_j, M_j, R \cap J_j \times M_j)$



(a) $(\{2\}, \{k\}, \emptyset)$ (b) Sous-contexte à gauche de Fig. 4.3 (c) $(\{9\}, \{b\}, \emptyset)$

FIGURE 4.2 : Les trois treillis facteurs de la décomposition, avec leur sous-contexte en légende

	k
2	↕

	c	d	f	g
3	↕	x	↓	x
5	x	x	↕	↕
6	x	↕	x	↓

	b
9	↕

FIGURE 4.3 : Les trois sous-contextes monogènes engendrés de gauche à droite par 2, 5, 9, du contexte de la table 2.3.

Considérons le contexte réduit de la table 4.1. Chaque itération de l'algorithme 1 est décrite dans la figure 4.1 : pour chaque valeur de j, l'entrée et la sortie de l'algorithme 2 sont données ainsi que les sous-contextes monogènes qui appartiennent à $\mathcal{L}$ à la fin du processus. Ainsi nous obtenons trois treillis facteurs (voir figure 4.2).

Le sous-contexte principal est celui du centre de la figure 4.3. Les deux autres sont $(\{2\}, \{k\}, \emptyset)$ et $(\{9\}, \{b\}, \emptyset)$. Ces deux derniers sont intéressants car :

- Ils montrent que le treillis de départ possède des parties distributives. En effet, ces deux sous-contextes contiennent exactement une double flèche par ligne et par colonne, ce qui est une propriété des treillis distributifs [1].
- Ils font apparaître une dichotomie : un concept contient 2 ou (exclusif) k ; de même pour 9 et b.
- Finalement, dans le contexte réduit, les flèches apportent une information complémentaire aux croix.

[1] Un treillis L est distributif, lorsque les opérations sup et inf le sont, l'une par rapport à l'autre.

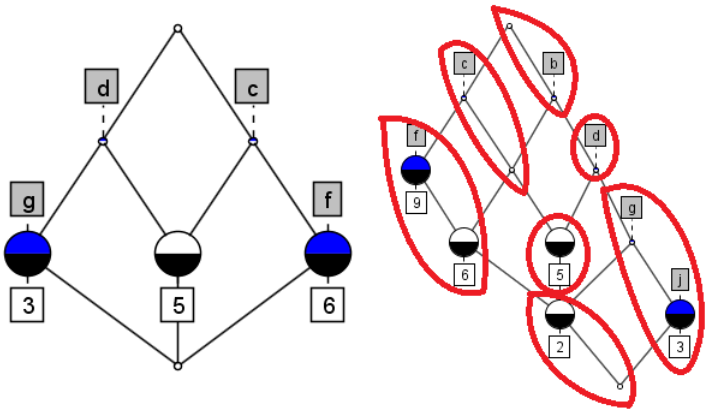


FIGURE 4.4 : Treillis quotient et sa relation de congruence

Le sous-contexte principal, au centre, est plus difficile à interpréter. Pour les deux précédents, nous avions une relation simple $2 \updownarrow k$ ou $9 \updownarrow b$, ce qui signifie que 2 et k sont presque des compléments ou des contraires. Heuristiquement, pour le contexte au centre, nous obtenons une sorte de relation multiple $(3, 5, 6) \updownarrow (c, d, f, g)$.

La figure 4.4 montre un couple treillis quotient et sa relation de congruence correspondante.

4.1.3 Morphisme injectif et FCA

La décomposition sous-directe d'un treillis L ayant pour facteurs $L_1, \ldots, L_n$ vient avec un morphisme injectif de L dans le produit direct $L_1 \times \cdots \times L_n$. Ce morphisme est précisé par la bijection entre les sous-contextes compatibles et les relations de congruences donnée dans les corollaires de la sous-section 4.1.1 :

Proposition 4.1.3. *Soit $(J, M, R \cap J \times M)$ un sous-contexte compatible d'un contexte (O, A, R), alors la relation $\Theta_{J,M}$ définie sur les couples de concepts (A_1, B_1), $(A_2, B_2) \in O \times A$ par :*

$$(A_1, B_1)\Theta_{J,M}(A_2, B_2) \iff A_1 \cap J = A_2 \cap J \iff B_1 \cap M = B_2 \cap M$$

est une relation de congruence, et son treillis quotient est isomorphe au treillis de concepts du sous-contexte $(J, M, R \cap J \times M)$.

L'algorithme 3 permet de calculer ce morphisme : chaque concept de L est transformé en un concept du treillis produit, puis ces derniers sont marqués. En notant Σ le nombre de noeuds du treillis initial et n le nombre de facteurs dans la décomposition, la complexité de cet algorithme est $\mathcal{O}(\Sigma n)$. A partir de

l'algorithme 3, on obtient le marquage du treillis produit, illustré par la figure 4.5.

Ce morphisme se calcule simplement en parcourant les noeuds du treillis initial, en les transformant en noeuds du treillis produit et en marquant ces derniers. On obtient le marquage du treillis produit, illustré par la figure 4.5. Evidemment, cet algorithme n'a pas vocation à être utilisé pour des applications réelles utilisant de grands contextes puisque le treillis produit est beaucoup plus gros que le treillis original et l'un des objectifs principaux de la décomposition est de travailler avec des treillis plus petits. Cet algorithme est uniquement utile à des fins de tests ou d'illustrations. Cependant, il peut être étendu pour un usage élémentaire en FCA. Une fois que la décomposition sous-directe d'un contexte réduit (O, A, R) en sous-contextes irréductibles $C_1, \ldots, C_n$ a été obtenue, une exploration interactive et un processus de fouille peuvent facilement être envisagés, en utilisant des traitements simples et en évitant la génération du treillis issus du contexte global (O, A, R).

- Calcul du plus petit concept de L contenant une liste d'objets ou d'attributs donnés, et identification de leur voisinage.
- Calcul du plus petit concept c_{ij} et de son voisinage dans certains treillis quotients, qui contiennent des objets ou attributs spécifiques. Chaque treillis L_i devenant une vue spécifique sur les données.

Algorithme 3 : Morphisme Injectif

Entrées : Treillis initial L;
Les sous-contextes (J_j, M_j, R_j);
Le treillis produit $P = L_1 \times \ldots \times L_n$
Output : Le treillis produit P dans lequel les noeuds de L ont été marqués.

```
1 pour tous les c = (A, B) ∈ L faire
2     pour tous les (J_j, M_j, R_j) faire
3         Calculer (A ∩ J_j, B ∩ M_j);
4     Marquer, dans P, le noeud-produit Π_j(A ∩ J_j, B ∩ M_j);
```

Enfin, l'algorithme 3 peut être réduit au calcul du morphisme dans un seul treillis facteur ; ainsi, il est intéressant pour identifier les noeuds qui sont dans la même classe pour la relation de congruence correspondante.

4.2 Implications et relations de congruence

Les relations de congruence ont été beaucoup étudiées, en particulier à travers le morphisme de projection. A partir des propriétés du treillis initial, des propriétés du treillis quotient sont déduites. Cependant, les problématiques inverses ont

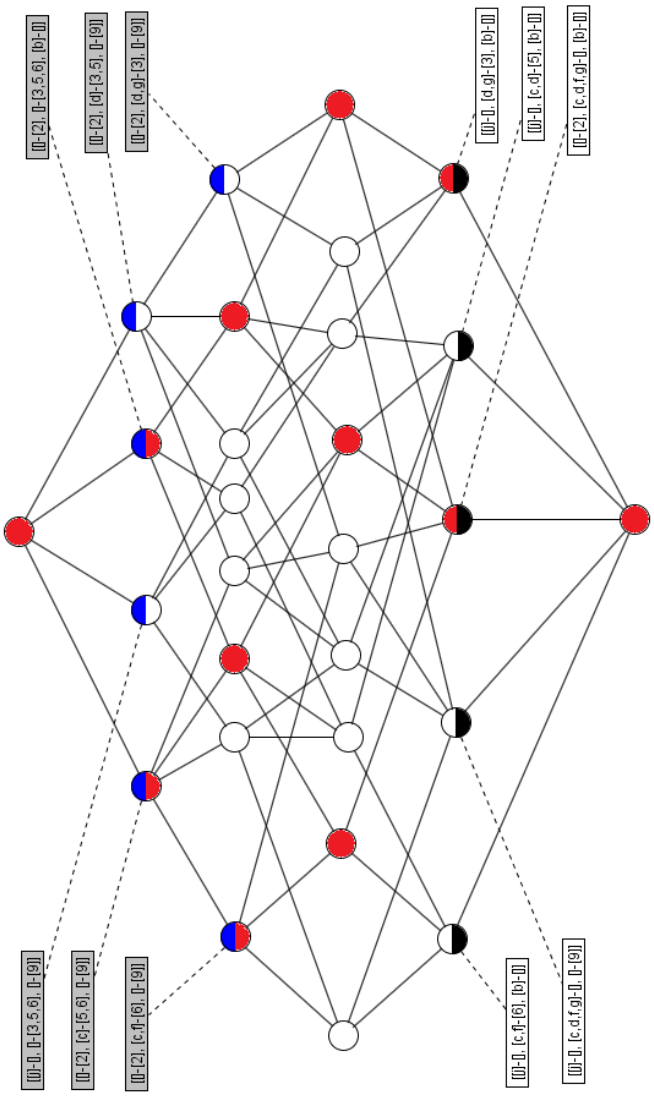


FIGURE 4.5 : Le treillis produit marqué. Les noeuds en rouge correspondent au treillis initial

Treillis initial			Treillis facteur		
d	$\rightarrow$	b			
f	$\rightarrow$	c	f	$\rightarrow$	c
g	$\rightarrow$	bd	g	$\rightarrow$	d
k	$\rightarrow$	bdg			
$bcdf$	$\rightarrow$	g	cdf	$\rightarrow$	g
$bcdg$	$\rightarrow$	f	cdg	$\rightarrow$	f

TABLE 4.2 : A gauche, la base canonique du treillis inital. A droite, la base canonique du treillis au centre de la figure 4.2

rarement été étudiées [140]. Plus précisément, si L est un treillis, on suppose qu'il est treillis quotient d'un autre treillis, $L = \tilde{L}/\Theta$. Quelles sont alors les propriétés de $\tilde{L}$ que l'on peut déduire à partir de celles de L ? Ces questions sont des problèmes de relèvement.

Dans cette section, nous souhaitons clarifier les liens entre les implications d'un treillis initial donné et celle d'un de ses treillis quotients. Plus précisement, étant donnée une implication ou une base d'implications dans le treillis initial (respectivement dans un treillis quotient), quelles implications peut-on déduire dans le treillis quotient (respectivement treillis initial) ?

Soit L un treillis de concepts déduit d'un contexte $K = (O, A, R)$ et Θ une relation de congruence sur L. Nous allons étudier les liens entre les implications de L et celles de L/Θ.

4.2.1 Exemple

Considérons le contexte de la table 2.3 et son treillis de concepts de la figure 2.3. Ce contexte peut être recouvert par trois sous-contextes monogènes (i.e. chaque observation et chaque attribut est dans au moins un sous-contexte). Ces sous-contextes sont donnés dans la figure 4.3 et les treillis de concepts correspondant sont dans la figure 4.2.

Aucun des treillis quotients engendrés par les objets 2 et 9 ne possède d'implication non-triviale. Nous nous focaliserons donc sur le treillis quotient engendré par l'objet 5. La base canonique du treillis initial et de ce treillis quotient sont donnés dans la table 4.2 [97, 86].

Soit $K = (O, A, R)$ un contexte, $(J, M, R \cap J \times M)$ avec $J \subseteq O$, $M \subseteq A$ un sous-contexte flèche-fermé, $S = R \cap J \times M$, Θ la relation de congruence correspondant à ce sous-contexte, L le trellis de concepts de K et L/Θ le treillis de concepts du sous-contexte flèche-fermé. Comme nous avons deux contextes (O, A, R) et (J, M, S) avec $S = R \cap J \times M$, nous utilisons les notations spéciales déjà mentionnées :

$$\forall X \subseteq O, X^R = \{a \in A, \forall x \in X, xRa\}$$

$$\forall X \subseteq J, X^S = \{a \in M, \forall x \in X, xSa\}$$

$$\forall Y \subseteq A, Y^R = \{o \in O, \forall y \in Y, oRy\}$$
$$\forall Y \subseteq M, Y^S = \{o \in J, \forall y \in Y, oSy\}$$

Ainsi, l'opérateur de fermeture pour X dans L est $X \mapsto X^{RR}$, tandis que dans L/Θ, il s'agit de $X \mapsto X^{SS}$. De manière duale, on peut définir $Y \mapsto Y^{RR}$ et $Y \mapsto Y^{SS}$. Maintenant, notre exemple peut être utilisé comme contre-exemple pour quelques questions :

- Rappelons qu'une implication $X \to \tilde{X}$, avec $\tilde{X} \subseteq M$ est informative dès que $\tilde{X} \not\subseteq X$, et bien sûr $X \to X^{RR}$ est informative lorsque X n'est pas fermé. D'un point de vue général, il peut arriver que $X \to X^{RR}$ soit informative alors que $X \to X^{SS}$ ne l'est pas. C'est effectivement le cas pour $X = \{d\}$ dans notre exemple. Cependant, il est important de noter que nous avons évidemment : si $X \to X^{SS}$ est informative, alors $X \to X^{RR}$ est informative. Cette observation est importante pour ce que nous souhaitons faire. En effet, nous souhaitons déduire des implications de L/Θ vers L, i.e., utiliser le morphisme $\pi : L \to L/\Theta$.

- Cette première observation a des conséquences fortes. Considérons la situation générale suivante : supposons que $K = (O, A, R)$ est un contexte couvert par une famille de sous-contextes flèche-fermés $(J_t, M_t, R \cap J_t \times M_t)$, où $O = \cup_{t \in T} J_t$ et $A = \cup_{t \in T} M_t$. A nouveau, une implication $X \to X^{RR}$ pourrait être informative, (par exemple $d \to b$), alors que l'on ne peut pas la trouver dans les sous-contextes. Cela signifie que des implications qui sont vraies dans le contexte initial peuvent ne pas être retrouvées dans les sous-contextes.

- En particulier, toujours en utilisant un recouvrement par des sous-contextes flèche-fermés, il pourrait être impossible de déduire la base canonique (ou la base canonique directe) à partir des sous-contextes recouvrants.

4.2.2 Résultats théoriques

Dans la suite, nous donnons une nouvelle formule qui établit un lien entre X^{RR} et X^{SS}. Alors, nous prouvons que si X est quasi-fermé dans K, alors $X \cap M$ est aussi quasi-fermé dans $(J, M, R \cap J \times M)$ avec une condition supplémentaire. Enfin, nous montrons que tout générateur de L/Θ est aussi un générateur de L.

Formule sur les concepts

Supposons que (X, Y) soit un concept de $K = (O, A, R)$ et (J, M, S) un sous-contexte compatible. Alors, par définition, $(X \cap J, Y \cap M)$ est un concept de (J, M, S). Ainsi d'un côté nous avons $X^R = Y, Y^R = X$ et de l'autre $(X \cap J)^S = Y \cap M$, $(Y \cap M)^S = X \cap J$. En combinant les deux, on obtient :

$$(X \cap J)^S = X^R \cap M,\ (Y \cap M)^S = Y^R \cap M$$

Soit X une partie quelconque de A, alors (X^R, X^{RR}) est un concept de (O, A, R) et la formule précédente donne :

$$X^{RR} \cap M = (X^R \cap J)^S,\ X^R \cap J = (X^{RR} \cap M)^S$$

En utilisant ces deux égalités, on obtient :

$$(*)\ X^{RR} \cap M = (X^R \cap J)^S = (X^{RR} \cap M)^{SS}$$

De plus, comme $X \subseteq X^{RR}$, il est également vrai que $(X \cap M)^{SS} \subseteq (X^{RR} \cap M)^{SS}$. Mais, par compatibilité de (J, M, S), nous avons $(X^{RR} \cap M)^{SS} = X^{RR} \cap M$. On peut conclure que :

$$(X \cap M)^{SS} \subseteq X^{RR} \cap M$$

Cependant, dans le cas général, il n'y a pas égalité. Par exemple, avec $X = \{j\}$, on déduit :

$$X^{RR} = \{b, d, g, j\},\ X^{RR} \cap M = \{d, g\},\ X \cap M = \emptyset,\ (X \cap M)^{SS} = \emptyset$$

Parallèlement, notre précédente formule peut être utilisée avec **une partie X de M**, et on obtient alors :

$$(**)\ X^{SS} \subseteq X^{RR} \cap M \subseteq X^{RR}$$

Nous pouvons maintenant prouver notre formule principale. Soit $X \subseteq M$. Avec $(*)$ et $(**)$, on a :

$$X^{SS} \subseteq X^{RR} \cap M = (X^{RR} \cap M)^{SS}$$

De plus, comme (J, M, S) est un sous-contexte, on a $X^S \subseteq X^R$. Mais $X^S \subseteq J$ et ainsi $X^S \subseteq X^R \cap J$. Alors $(X^R \cap J)^S \subseteq X^{SS}$. Or $(X^R \cap J)^S = X^{RR} \cap M$, donc $X^{RR} \cap M \subseteq X^{SS}$. Finallement, nous avons :

$$X^{RR} \cap M \subseteq X^{SS} \subseteq X^{RR} \cap M$$

Par conséquent, si X est une partie quelconque de M, on a :

$$X^{RR} \cap M = X^{SS}$$

Nous avons donc le résultat suivant :

Théorème 4.2.1. *Pour toute partie $Y \subseteq A$, on a :*

$$(*)\ Y^{RR} \cap M = (Y^{RR} \cap J)^{SS}$$

Et pour tout partie $Y \subseteq M$, on a :

$$Y^{RR} \cap M = Y^{SS}$$

Notons $\pi : L \to L/\Theta$ le morphisme canonique. Du point de vue des treillis de concepts, ce morphisme est $\pi(B, C) = (B \cap J, C \cap M)$. Nous pouvons également étendre ce morphisme aux implications, de la manière suivante :

$$\pi(Z \to Z^{RR}) = Z \cap M \to (Z \cap M)^{SS}$$

On rappelle également [88] que tout concept (U, V) de $K = (O, A, R)$ tel que $(B \cap J)^{RR} \subseteq U \subseteq (C \cap M)^R$ ou tel que $(B \cap J)^R \subseteq V \subseteq (C \cap M)^{RR}$, vérifie :

$$\pi(U, V) = \pi(B, C) = (B \cap J, C \cap M)$$

On peut également affirmer que $Y^R \cap J \subseteq (Y \cap M)^S$. En effet, si $o \in Y^R \cap J$, alors $o \in J$ et $o \in Y^R$. La dernière signifie que pour tout $y \in Y$, oRy. Ainsi, en particulier, pour tout $y \in Y \cap M$, oRy. Cependant $J = R \cap J \times M$, donc de $o \in J$, $y \in M$ on obtient $oRy \Longrightarrow oSy$. Ainsi on peut conclure que $Y^R \cap J \subseteq (Y \cap M)^S$.

Soit $T \subset Z \subset M$ tel que $T^{RR} = Z^{RR}$, alors $T^{SS} = Z^{SS}$. En effet, nous avons évidemment $T^{SS} \subseteq Z^{SS}$ puisque $\bullet^{SS}$ est un opérateur de fermeture. De plus, avec $(*)$ et $T \subseteq Z \subseteq M$, nous avons :

$$Z^{SS} \subseteq Z^{RR} \cap M = T^{RR} \cap M = T^{SS}$$

Quasi-fermés

On rappelle que $X \subseteq A$ est quasi-fermé si $X^{RR} \neq X$ et si :

$$\forall Y \subseteq X,\ Y^{RR} \subset X \text{ ou } Y^{RR} = X^{RR}$$

Les pseudo-intents sont minimaux parmi les quasi-fermés, c'est à dire que, X est un pseudo-intent s'il est quasi-fermé et qu'il vérifie :

$$\forall Y \subset X \text{ avec } Y \text{ quasi-fermé et } Y^{RR} = X^{RR},\ \text{alors } Y = X$$

Montrons que la propriété d'être quasi-fermé est compatible avec les sous-contextes compatibles, c'est à dire que si $X \subseteq A$ satisfait :

$$\forall Y \subseteq X,\ Y^{RR} \subseteq X \text{ ou } Y^{RR} = X^{RR}$$

alors $X \cap M$ satisfait :

$$\forall Y \subseteq X \cap M,\ Y^{SS} \subseteq X \cap M \text{ ou } Y^{SS} = (X \cap M)^{SS}$$

En effet, supposons que $Y \subseteq X \cap M$, alors $Y^{SS} \subseteq (X \cap M)^{SS} = X^{RR} \cap M$ et $Y^{SS} = Y^{RR} \cap M$. Par conséquent, $Y^{RR} \cap M \subseteq X^{RR} \cap M$.

Cependant, si on suppose que $X^{RR} = Y^{RR}$ ou $Y^{RR} \subseteq X$, alors dans le premier cas, on obtient $Y^{SS} = Y^{RR} \cap M = X^{RR} \cap M = X^{SS}$ alors que dans le second, on obtient $Y^{SS} = Y^{RR} \cap M \subset X \cap M$.

Nous avons donc le théorème suivant :

Théorème 4.2.2. *Si X est quasi-fermé dans $K = (O, A, R)$ et si $(X \cap M)$ n'est pas fermé dans (J, M, S), alors $X \cap M$ est quasi-fermé où $J \subseteq O$, $M \subseteq A$ et $S = R \cap J \times M$.*

La réciproque n'est en général pas vraie. Par exemple, l'ensemble $\{c, d, g\}$ est un pseudo-intent dans le sous-contexte précédemment utilisé mais il n'est même pas quasi-fermé dans le contexte initial.

De plus, le théorème 4.2.2 n'est en général pas vrai pour les pseudo-intents car la propriété de minimalité peut être perdue à l'intérieur du sous-contexte.

Générateurs minimaux

Supposons que $Z \to Z^{SS}$ est une implication de la base canonique directe de (J, M, S), ce qui signifie que Z est un générateur minimal. Nous voulons prouver que $Z \to Z^{SS}$ est aussi une implication de la base canonique directe de $K = (O, A, R)$. Cela signifie que si Z est un générateur minimal dans L/Θ, alors il l'est aussi dans L. Autrement dit, s'il existe $T \subseteq Z$ tel que $T^{RR} = Z^{RR}$, alors $Z = T$. En effet, nous avons montré que si $T \subseteq Z$ tel que $T^{RR} = Z^{RR}$, alors $T^{SS} = Z^{SS}$, et, comme Z est un générateur minimal dans (J, M, S), on obtient $Z \subseteq T$ et donc $Z = T$.

Théorème 4.2.3. *Si Z est un générateur minimal de L/Θ, il en est de même dans L.*

La mauvaise nouvelle pour notre étude, qui constitue cependant un résultat, peut être résumée de la manière suivante : Si $X \to X^{RR}$ est une implication informative de $K = (O, A, R)$, mais si $X \not\subseteq M$ ou $X^{RR} \not\subseteq M$, il peut arriver que $X \to X^{SS}$ ne soit pas informative dans (J, M, S).

Nous prévoyons d'étudier les problèmes suivants que nous n'avons pas abordés ici :

- Etant donnée une implication $Z \to Z^{RR}$ de la base canonique de $K = (O, A, R)$ telle que l'ensemble $Z \cap M$ ne soit ni vide, ni fermé, est-ce que $Z \cap M \to (Z \cap M)^{SS}$ est dans la base canonique de (J, M, S) ?
- Plus généralement, étant donnée la base canonique B (resp. directe) de K, est-ce que le sous-ensemble des implications informatives de $\pi(B)$ est la base canonique (resp. directe) de (J, M, S) ?
- Peut-on exploiter un sous-contexte, directement, sans générer d'implications, pour produire des implications valides dans le contexte initial ?
- Peut-on déduire un algorithme efficace de calcul de l'opérateur de fermeture à partir utilisant ce théorème ?

Remarquons que dans cette section, nous nous sommes focalisés sur les bases canoniques et canoniques directes. Cependant, mentionnons la D-base [3, 125] qui est particulièrement bien adaptée à ce procédé de récupération utilisant des

congruences. En effet, nous avons mentionné une dernière équivalence, entre les relations de congruence et les ensembles D-fermés définis par [93, 76] :

$$a\ D\ b,\ a \in X \Longrightarrow b \in X$$

Alors, dans ce cas, tout se passe pour le mieux ; si le treillis initial est recouvert par des ensembles D-fermés, alors la D-base peut être récupérée (sauf sa partie binaire). Toujours concernant la D-base, il existe également une autre méthode de calcul pour les grands jeux de données [1].

Dans cette section, nous avons utilisé les treillis quotient par une relation de congruence afin de calculer des implications. Dans la section suivante, nous utilisons à nouveau ces relations de congruence, mais cette fois pour introduire une nouvelle décomposition.

4.3 Construction de doublement de convexe inversée

Dans cette section, nous décrivons la construction de doublement inversée qui utilise des relations de congruence.

Etant donné un treillis L, rappelons que nous cherchons un treillis L_C contenant un convexe C tel $L = L_C[C]$ où $L_C[C]$ désigne le treillis obtenu à partir de L_C et C par la construction de doublement de convexe.

Un exemple détaillé de cette construction est donné à la section 6.2.2.

4.3.1 Principales étapes

Avant de donner tous les détails à propos de cette décomposition, commençons par donner les étapes principales de notre construction.

Etapes principales de la preuve de la validité de notre construction

Etant donné un treillis L, nous cherchons un treillis L_C et un convexe $C \subset L_C$ tel que L soit isomorphe à $L_C[C]$.

1. Nous utilisons tout d'abord le théorème 2.3.11 de Day [49]. Avec ce théorème 2.3.11, nous savons que les bons candidats sont obtenus comme treillis quotients engendrés par une relation de congruence qui est un atome.
   Il est important de noter que nous obtenons seulement de bons candidats car nous n'utilisons pas l'hypothèse de treillis normalement congruent. Nous n'avons d'ailleurs pas défini cette notion car nous n'en avons pas besoin.

2. Ensuite nous utilisons l'équivalence du théorème 2.3.6 dont la preuve peut être trouvée dans [88]. D'après ce théorème 2.3.6, nous savons que les relations de congruence correspondent aux sous-contextes flèche-fermés.

3. Nous utilisons alors le théorème de Geyer 2.3.10. Il nous reste alors seulement à vérifier que nos bons candidats vérifient la condition de Geyer 2.9.

$$R \cap ((O \backslash J) \times (A \backslash M)) = \emptyset \tag{4.1}$$

4. Si aucun sous-contexte n'est valide, la décomposition n'est pas possible. Sinon, chacun des sous-contextes génère un treillis L_C qui convient. Il y a donc une dichotomie : soit aucun sous-contexte ne convient et on ne fait pas cette décomposition, soit un ou plusieurs sous-contextes conviennent et il est possible de choisir un quelconque de ces sous-contextes pour décomposer le treillis d'une manière.

Etapes principales de la construction.

Les principales étapes sont les suivantes :

1. La première étape de notre construction consiste à calculer les atomes du treillis des relations de congruence avec l'algorithme 2.

2. Ensuite, nous sélectionnons les sous-contextes flèche-fermés qui satisfont la condition de Geyer, avec une complexité en $\mathcal{O}(|O|\,|A|)$. Chacun de ces sous-contextes engendre un treillis L_C qui convient.

3. La dernière étape consiste à trouver le convexe C dans L_C, avec une complexité en $\mathcal{O}(\Sigma_{L_C})$ où Σ_{L_C} est le nombre de noeuds de L_C. Remarquons qu'il existe déjà un autre algorithme de reconnaissance de convexe [24].

4.3.2 Le treillis des relations de congruence

On suppose donné le contexte réduit (O, A, R) d'un treillis L.

Rappelons qu'une relation de congruence est une relation d'équivalence compatible avec la structure du treillis. En tant que cas particulier de relation binaire, l'ensemble des relations de congruence hérite d'une structure de treillis : c'est un sous-treillis du treillis des relations binaires, ordonnées par l'inclusion. Avec le théorème 2.3.6, ce treillis est aussi le treillis des sous-contextes flèche-fermés. Comme expliqué précédemment, nous cherchons les atomes de ce treillis. A cet effet, nous introduisons un nouveau contexte, à savoir le contexte des sous-contextes flèche-fermés, qui se définit et se calcule comme suit. On calcule tout d'abord les sous-contextes monogènes.

Rappelons la définition (4.1.2) de contexte monogène :

Définition 4.3.1. Un contexte (J, M, R) est dit monogène s'il peut être obtenu par flèche-fermeture à partir d'un seul $j \in J$. Ainsi (J, M, R) est monogène s'il est le plus petit sous-contexte flèche-fermé contenant $j \in J$.

L'ensemble de ces sous-contextes monogènes contient l'ensemble des sous-contextes flèche-fermés sup-irréductibles du treillis des sous-contextes flèche-fermés. Ainsi, l'ensemble des sous-contextes monogènes engendre le treillis des sous-contextes flèche-fermés. Ces sous-contextes flèche-fermés sont stockés en tant qu'observations d'un nouveau contexte de la manière suivante : un sous-contexte flèche-fermé est caractérisé par son ensemble d'attributs, puisqu'il est fermé. Ainsi notre nouveau contexte possède les mêmes attributs A que le premier et puisque les sous-contextes flèche-fermés sont calculés en fermant une observation $j \in O$, ils peuvent être identifiés avec ce j. L'ensemble des observations du nouveau contexte est donc O. Finalement, il y a une croix dans le nouveau contexte entre un objet j et un attribut m si et seulement si le sous-contexte monogène engendré par j contient l'attribut m.

A partir de ce contexte, on peut déduire le treillis des sous-contextes flèche-fermés et en particulier ses atomes.

4.3.3 Le treillis quotient

Nous possédons désormais une liste de sous-contextes flèche-fermés qui correspondent aux relations de congruences qui sont des atomes.

La seconde étape consiste à enlever de cette liste les contextes suivant :

- Le contexte vide. Dans le cas particulier où le treillis initial possède seulement deux relations de congruence (treillis simple), il n'y a qu'un seul atome, à savoir l'élément maximal. Nous excluons ce cas qui donne une décomposition triviale.

- Tout treillis ne satisfaisant pas à la condition de Geyer.

Après ce processus d'élimination, il est possible que notre liste soit vide. Comme la condition de Geyer est nécessaire et suffisante, nous pouvons conclure que le treillis original ne peut pas être réduit par la construction de doublement inversée.

A l'opposé, tout contexte figurant encore dans la liste génère un treillis L_C pour lequel il existe un convexe C tel que le treillis initial L vérifie $L = L_C[C]$.

On supposera dans la suite qu'un tel contexte a été choisi ; ce choix fixe de manière unique L_C et C. Le treillis L_C est le treillis de concepts de ce contexte. La sous-section suivante explique comment déterminer C.

4.3.4 Trouver le convexe

Etant donné un treillis L_C satisfaisant aux conditions de la sous-section 4.3.3, la dernière étape consiste à identifier dans L_C les noeuds du convexe C tel que $L = L_C[C]$.

Avec le théorème 2.3.10 qui donne une condition nécessaire et suffisante, nous savons déjà que L est obtenu par la construction de doublement à partir de L_C. Ainsi, chaque concept de L_C donne naissance à un seul concept de L si

et seulement s'il n'est pas dans C et à deux concepts de L si et seulement s'il est dans C.

Soyons plus précis à propos de ces deux concepts, et tout d'abord rappelons quelques notations : (O, A, R) est le contexte réduit du treillis L. On considère $(J, M, R \cap J \times M)$ un atome dans le treillis des sous-contextes flèche-fermés qui vérifie la condition de Geyer. En particulier, $(J, M, R \cap J \times M)$ est un sous-contexte de (O, A, R), donc $J \subseteq O$ et $M \subseteq A$. A partir de ce contexte, on construit le treillis L_C duquel le convexe C a été doublé. Soit c un concept de L_C. D'après [88], on sait également que les sous-contextes flèche-fermés sont aussi les sous-contextes compatibles. Ainsi $(J, M, R \cap J \times M)$ est un sous-contexte compatible. Avec ce résultat, c peut être écrit $c = (H \cap J, N \cap M)$ où (H, N) est un concept de L. Comme c est un concept, nous avons $(H \cap J)' = N \cap M$ et $(N \cap M)' = H \cap J$. Par ailleurs, nous avions noté ces opérations $\bullet^L$ dans le contexte réduit (O, A, R) de L ; ainsi nous avons aussi $H^L = N$ et $N^L = H$. Maintenant, nous pouvons conclure que, si $c = (H \cap J, N \cap M)$ n'est pas dans le convexe, il vient d'un unique concept de L, à savoir (H, N) et donc nous devons avoir $((H \cap J)^{LL}, (H \cap J)^L) = (H, N) = ((N \cap M)^L, (N \cap M)^{LL})$. Dans l'autre cas, nous avons $((H \cap J)^{LL}, (H \cap J)^L) \neq ((N \cap M)^L, (N \cap M)^{LL})$.

A partir de là, nous pouvons établir une condition sur un concept pour qu'il soit ou non dans le convexe :

Théorème 4.3.1. *Un concept $c = (H \cap J, N \cap M)$ est dans le convexe si et seulement si :*

$$(H \cap J)^L \subseteq (N \cap M) \text{ ou } (N \cap M)^L \subseteq (H \cap J) \tag{4.2}$$

Démonstration. Nous avons déjà vu que si c est dans le convexe alors :

$$((H \cap J)^{LL}, (H \cap J)^L) = (H, N) \tag{4.3}$$
$$= ((N \cap M)^L, (N \cap M)^{LL}) \tag{4.4}$$

Donc les deux inclusions sont vraies et sont même des égalités.

Supposons, réciproquement, que nous avons la première :

$$(H \cap J)^L \subset (N \cap M) \tag{4.5}$$

Le deuxième cas s'obtient par dualité.

Tout d'abord, nous prouvons que cette inclusion est en fait une égalité. En effet, comme $(J, M, R \cap J \times M)$ est un sous-contexte de (O, A, R), nous avons :

$$(H \cap J)' \subset (H \cap J)^L \tag{4.6}$$

Cependant $(H \cap J)' = (N \cap M)$, donc :

$$(N \cap M) \subset (H \cap J)^L \tag{4.7}$$

et dualement :

$$(H \cap J) \subset (N \cap M)^L \tag{4.8}$$

Ainsi, on déduit que $(H \cap J)^L = (N \cap M)$, et alors $(N \cap M)^L = (H \cap J)^{LL}$. De plus :

$$(H \cap J) \subset (N \cap M)^L \Longrightarrow (N \cap M)^{LL} \subset (H \cap J)^L \tag{4.9}$$

et avec $(N \cap M) = (H \cap J)^L$, on déduit que :

$$(N \cap M) = (N \cap M)^{LL} \tag{4.10}$$

puisque $(N \cap M) \subset (N \cap M)^{LL}$.

Maintenant de $(H \cap J)^L = (N \cap M) = (N \cap M)^{LL}$ et $(N \cap M)^L = (H \cap J)^{LL}$, on obtient :

$$((H \cap J)^L, (H \cap J)^{LL}) = ((N \cap M)^{LL}, (N \cap M)^L) \tag{4.11}$$

Cela signifie que le concept $((H \cap J), (N \cap M))$ du petit treillis L_C qui satisfait la condition $(H \cap J)^L \subset (N \cap M)$ ou $(N \cap M)^L \subset (H \cap J)$ vient d'un unique concept du treillis L, à savoir le concept :

$$((H \cap J)^{LL}, (H \cap J)^L) = (H, N) \tag{4.12}$$

$$= ((N \cap M)^L, (N \cap M)^{LL}) \tag{4.13}$$

Réciproquement, si $((H \cap J), (N \cap M))$ ne satisfait pas à la condition précédente, il vient de deux concepts distincts de L :

$$((H \cap J)^{LL}, (H \cap J)^L) \neq ((N \cap M)^L, (N \cap M)^{LL}) \tag{4.14}$$

Ainsi pour obtenir L à partir de L_C, il suffit de doubler ces concepts et ces concepts sont donc exactement ceux du convexe cherché. □

Un exemple de telle décomposition est donné dans la section 6.2.2.

4.4 Relations de tolérance complètes

Commençons par rappeler les points généraux sur les tolérances complètes. Les relations de tolérance complètes sont des relations réflexives et symétriques compatibles avec la structure de treillis. Elles apparaissent donc comme une généralisation des relations de congruence puisque l'axiome de transitivité est omis. Le qualificatif "complète" sera omis par la suite.

Ces relations de tolérance sont particulièrement intéressantes pour de nombreuses raisons :

- Elles peuvent être interprétées comme des relations de similarité ; ainsi, elles sont porteuse d'une sémantique lorsque l'on étudie des jeux de données.

- En tant que généralisation des relations de congruence, elles sont plus nombreuses ; il est donc plus facile de les utiliser sur des données.

- Etant donné un treillis L et une relation de tolérance τ, il est possible de définir un ensemble quotient L/τ et une structure de treillis dessus ; une structure de treillis sur le quotient est donc conservée, il est donc toujours possible d'utiliser ce treillis quotient pour de la classification. Ainsi, lorsque l'on étudie un treillis L, il est possible de réduire sa taille en considérant L/τ, pour une relation de tolérance τ qui nous intéresse.

- Les relations de tolérance apparaissent aussi comme un ingrédient essentiel de la décomposition *Atlas* qui est une autre décomposition de treillis.

- En tant que cas particulier de relation binaire, l'ensemble des relations de tolérance peut être ordonné par l'inclusion ; l'ensemble ordonné obtenu est un treillis, c'est le treillis des relations de tolérance. De nombreux résultats théoriques à propos de ce treillis [95] sont connus et peuvent être utilisés pour étudier des jeux de données.

Notons également que les relations de tolérance non complètes (*i.e.* seulement réflexive et symétrique mais non compatibles avec les lois du treillis) sont utilisées en lien avec la FCA [104, 129].

Ce treillis des relations de tolérance est l'espace de navigation utilisé par notre projet LattExp (chapitre 3 et section 6.1). Dans cette section, nous présentons les algorithmes permettant de calculer des tolérances et de calculer le treillis de toutes les tolérances.

L'algorithme permettant de calculer une tolérance est un algorithme de fermeture naïf ; sa complexité dans le pire des cas est exponentielle en le minimum de $|O|$ et $|A|$. Il est basé uniquement sur la définition 2.3.29 des relations blocs qui est une notion équivalente à celle de tolérance (théorème 2.3.8). Etant donné un contexte $K = (O, A, R)$ initial et une autre relation binaire S contenant R, i.e. $R \subseteq S$, il faut que pour tout $o \in O$, o^S soit une intension de K. Or l'intension de K correspondante est o^{SRR}, donc lorsque l'inclusion $o^S \subset o^{SRR}$ est stricte, on ajoute dans la ligne o de S les croix de manière à avoir égalité. Les attributs sont traités de même et le processus est itéré jusqu'à convergence. Cet algorithme se termine car S est majoré par $O \times A$; d'autre part, comme nous raisonnons par condition nécessaire, la sortie est la plus petite tolérance contenant S. On obtient ainsi l'algorithme 4, dont les entrées-sorties seront précisées ultérieurement en fonction de l'usage qui en est fait.

Cet algorithme va maintenant être utilisé pour calculer tout le treillis des tolérances. Avec le théorème 2.3.9, les irréductibles sont connus. De plus, on a vu (section 2.3.9) que deux concepts (C_1, B_1), (C_2, B_2) sont en relation par S si et seulement si :

$$C_1 \times B_2 \cup C_2 \times B_1 \subseteq S$$

L'algorithme procède alors comme suit :

Algorithme 4 : Algorithme de calcul d'une relation de tolérance.

```
1 tant que S est modifié faire
2     pour tous les o observation faire
3         si o^S ⊊ o^SRR alors
4             Remplacer la ligne o^S par o^SRR;
5     pour tous les a attribut faire
6         si a^S ⊊ a^SRR alors
7             Remplacer la colonne a^S par a^SRR;
```

- Un contexte $K = (O, A, R)$ est donné en entrée correspondant à un treillis L. En sortie, un contexte TOL est retourné, dont le treillis de concepts est le treillis des tolérances de L.
- Les concepts de L sont parcourus et pour chacun de ces concepts, ses successeurs immédiats sont parcourus. De sorte que l'on a alors un couple (C_1, B_1), (C_2, B_2) de deux concepts consécutifs.
- A partir de ce couple, on forme le contexte suivant :

$$S = R \cup C_1 \times B_2 \cup C_2 \times B_1$$

  De sorte que nos deux concepts sont mis en relation.
- L'algorithme 4 est utilisé pour fermer S. À la sortie, S contient le contexte correspondant à la plus petite tolérance mettant (C_1, B_1) et (C_2, B_2) en relation.
- Cette tolérance S est ajoutée en tant qu'observation dans le contexte TOL.

On obtient ainsi l'algorithme 5.

Algorithme 5 : Algorithme de calcul du treillis des tolérances.

Entrées : Un contexte (O, A, R) de treillis de concepts L
Sorties : TOL : contexte dont le treillis de concepts est le treillis des tolérances de L

```
1 pour tous les (C1, B1) concept de L faire
2     pour tous les (C2, B2) successeur immédiat de (C1, B1) faire
3         Créer S = R ∪ C1 × B2 ∪ C2 × B1;
4         Fermer S avec l'algorithme 4;
5         Ajouter S comme observation de TOL;
```

Notons que dans la pratique cet algorithme est coûteux puisqu'il fait appel à l'algorithme 4 dont la complexité dans le pire des cas est exponentielle.

Il permet d'appréhender globalement l'espace de navigation, ce qui n'est pas indispensable. En effet, la borne inférieure de deux tolérances est l'intersection et la borne supérieure de deux tolérances s'obtient en fermant l'union. Ainsi avec ces deux opérations, on peut naviguer dans le treillis des tolérances sans le calculer complètement [2]. Cette approche est illustrée par un exemple à la section 6.1.

[2]Cette approche est également utilisée dans l'étude des dépendances fonctionnelles ; elle pourrait donc également être utilisée pour le calcul de bases d'implications.

Chapitre 5

Usage de MapReduce avec Hadoop

Sommaire

Pour des raisons préalablement soulignées, la FCA ne peut pas être utilisée pour améliorer les outils des données massives[1]. Cependant, lorsque l'on utilise un contexte massif au départ, les outils des données massives peuvent être utilisés pour gérer l'explosion du nombre de concepts ou de règles d'association. Ainsi, dans ce chapitre, nous présentons des calculs pour la FCA basés sur Hadoop.

L'idée directrice de ce chapitre est la suivante : On effectue un seul job MapReduce et l'on regarde ce que l'on arrive à obtenir à la sortie. Il ne s'agit pas d'obtenir un résultat complet et optimal mais, dans l'esprit des données massives, de faire simple et de passer à l'échelle.

Avant, commençons par une citation de Doug Cutting, le créateur du projet Hadoop, donnant l'explication suivante quant à l'origine du nom [178] :

[1] La FCA peut par contre apporter un point de vue complétementaire.

> The name my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless, and not used elsewhere : those are my naming criteria. Kids are good at generating such. Googol is a kid's term.

Le nom de notre projet, LattExp, ne répond malheureusement pas à ces critères. Nous avons choisi Hadoop pour les raisons suivantes :

- Dévoilé par Google en 2004, le paradigm MapReduce a montré son efficacité pour le calcul distribué.
- Hadoop est un projet open source mature ; il est robuste et très répandu. Un grand nombre de projets basés sur Hadoop existent[2].

Des travaux utilisant MapReduce pour les algorithmes de l'analyse des concepts formels existent [112, 190, 169]. L'objectif de ces travaux est de calculer complètement le treillis de concepts soit en mettant en place un mécanisme de MapReduce [112], soit en utilisant une implémentation existante, Twister [190]. Les travaux présentés ici se situent dans la même direction et visent à revisiter certains algorithmes de la FCA, en utilisant également [190] une implémentation existante de MapReduce, à savoir Hadoop. A la différence de Twister, Hadoop n'est pas orienté vers des algorithmes itératifs, de sorte que nous présenterons des algorithmes dans l'esprit "one-pass". Plus précisement, nous essayons d'obtenir autant de résultats que possible à l'aide d'une unique tâche MapReduce.

Dans les sections suivantes, nous présentons le paradigme MapReduce et en particulier son implémentation dans Hadoop. Puis dans la suite du chapitre, nous proposons des calculs liés à la FCA utilisant Hadoop :

- Dans la littérature, l'opérateur de fermeture est calculé de manière classique. Nous présentons ici une tâche MapReduce calculant juste la fermeture d'un ensemble d'attributs. Ce type de calcul peut être utile lorsque le contexte initial est tellement grand que même le calcul d'un seul concept s'avère difficile.
- Ensuite nous donnons un algorithme permettant de calculer les prédécesseurs d'un concept donné. Il est important de noter que le calcul des prédécesseurs requiert uniquement que les données soient lues, pas modifiées.
- Enfin en ce qui concerne les concepts, nous présentons un calcul de tous les concepts en seulement une tâche MapReduce. L'inconvénient est que cette tâche génère une quantité exponentielle de données pendant le calcul mais par ailleurs ces données sont distribuées sur le cluster et donc chaque noeud n'en possède qu'une quantité limitée. Enfin, remarquons que cet algorithme est tout à fait naïf ; il mériterait d'être amélioré.

Ensuite, des calculs concernant les implications sont présentés :

[2] `http://wiki.apache.org/hadoop/PoweredBy`

- Une première tâche MapReduce permet de calculer quelques implications de la base canonique directe. Cette tâche peut être incluse dans un processus itératif permettant alors de calculer toute la base canonique directe [3].

- Une deuxième tâche calcule un ensemble d'implications contenant la base canonique directe. Cependant cette tâche génère une quantité exponentielle de données, qui à nouveau sont distribuées sur le cluster.

5.1 Présentation générale du paradigme MapReduce

Nous présentons ici le paradigme MapReduce général ainsi que quelques spécificités techniques de son implémentation dans Hadoop que nous utiliserons. En premier lieu, le contexte, supposé massif, est stocké sur un système distribué, en l'occurrence HDFS (Hadoop Distributed File System). Les quatre étapes suivantes décrivent ce qui est appelé un job MapReduce.

1. Étape de séparation (SPLIT) : les données sont séparées et distribuées sur le cluster. De manière équivalente, une partition des lignes/observations du contexte est créée. Chaque noeud qui va effectuer une tâche d'association (MAP), a accès à une partie des données, i.e. une classe de la partition. Il est important de noter que chaque noeud n'a qu'un accès partiel aux données.

2. Étape d'association (MAP) : Cette phase consiste à lire ces données distribuées et à mettre en place une association clé-valeur $< Key, Value >$ à partir d'elles. Notons qu'une "clé" au sens de MapReduce, n'est pas une clé au sens des bases de données ; dans le cadre de MapReduce, c'est un identifiant permettant de regrouper plusieurs valeurs associées à cette même clé.

3. Étape de combinaison (COMBINE) : Cette phase intermédiaire peut être insérée entre Map et Reduce. Pour chaque noeud ayant effectué l'étape MAP, les données qu'il a produites sont traitées, en amont, par l'étape REDUCE. Ainsi chaque noeud concerné par la phase de réduction recevra des données pour lesquelles le travail est déjà partiellement effectué.

4. Étape de réduction (REDUCE) : chaque noeud en charge d'une opération de réduction reçoit toutes les données correspondant à une clé. Cela signifie qu'un tel noeud peut recevoir ou non plusieurs clés, mais pour chaque clé qu'il reçoit, il dispose de toutes les données associées à cette clé. Cette étape de réduction est celle qui effectue les calculs sur les paquets de données associés à une même clé.

[3] Cette approche est présentée rigoureusement à la section 5.3.

	a	b	c	d	e
1	x	x			
2		x	x	x	
3				x	x
4			x		
5	x		x		
6	x	x	x	x	x

TABLE 5.1 : Contexte utilisé pour illustrer l'approche MapReduce de la fermeture.

Hadoop offre également une fonctionnalité de cache distribué, permettant qu'une donnée, qui doit être petite, soit accessible en lecture à toutes les entités Map et Reduce. C'est la seule connaissance globale à laquelle les noeuds ont accès.

Dans la section suivante, ces différentes étapes sont illustrées à travers un algorithme de fermeture.

5.2 Calculs de concepts

5.2.1 Algorithme de fermeture d'un ensemble d'attributs

L'algorithme de fermeture est très simple mais il fait intervenir tous les mécanismes présentés précédemment. En entrée, on dispose d'un contexte massif stocké sur un système distribué. On souhaite obtenir la fermeture d'un ensemble donné d'attributs, noté A. Cet ensemble est placé dans le cache distribué de sorte que tous les mappers (noeuds réalisant une opération MAP) vont pouvoir le lire. Le premier point, lourd de conséquences, est que l'algorithme n'utilise pas de clé ; plus précisément il utilise une clé unique ayant la valeur "null". Ceci a pour première conséquence qu'il y aura un seul Reducer (noeud réalisant une opération REDUCE) qui va donc constituer un goulot d'étranglement. Cependant, en utilisant un Combiner (noeud réalisant une opération COMBINE), le Reducer ne recevra qu'une seule donnée par Mapper, ce qui rend l'algorithme efficace.

Exemple 5.2.1. Pour illustrer les différentes étapes de ce job MapReduce, nous utiliserons le contexte de la table 5.1

L'algorithme du Mapper est donné par l'algorithme 6.

Exemple 5.2.2. Pour notre exemple, supposons qu'il y ait 2 mappers. Lors de l'étape SPLIT, le premier Mapper reçoit les trois premières lignes et le second la suite. Si nous voulons obtenir la fermeture de $A = \{d\}$, alors avec l'algorithme de la figure 6, le premier Mapper crée les associations : $(null, bcd), (null, de)$. Quant au second, il crée l'association : $(null, abcde)$.

Algorithme 6 : Algorithme de l'étape MAP pour la fermeture de A.

```
pour tous les o observation faire
    si A ⊂ o' alors
        // L'observation possède tous les attributs de A;
        Créer l'association clé-valeur (null, o');
```

A la sortie des Mappers, on dispose donc de toutes les observations contenant tous les attributs de A [4].

Le Combiner et le Reducer effectuent l'algorithme 7.

Algorithme 7 : Algorithme de l'étape REDUCE (ou COMBINE) pour la fermeture.

```
C ← ∅;
// Ensemble d'attributs à produire;
pour tous les (null, O') couple clé-valeur reçu faire
    si (null, O') est le premier couple alors
        C ← O';
    sinon
        C ← O' ∩ C;
        // Revient à faire un "et" binaire;
Créer l'association (null, C);
```

Le Reducer ou Combiner produit un "et" binaire de toutes les observations contenant tous les attributs donnés en entrée ; c'est à dire qu'il calcule exactement la fermeture de cet ensemble d'attributs.

Exemple 5.2.3. Le Combiner agissant à la sortie du premier Mapper, effectue le "et" binaire sur les 2 couples reçus puis retourne $(null, d)$. Le second Combiner ne fait rien. Ensuite, le Reducer reçoit toutes les valeurs combinées de la clé $null$, c'est à dire $(null, d)$ et $(null, abcde)$ puis effectue la réduction pour retourner $(null, d)$.

L'opérateur "et" étant associatif, il revient au même de faire un "et" global que de faire un "et" sur des données partielles puis refaire un "et" sur les résultats obtenus. C'est cette propriété qui assure que l'approche précédente est valide.

Remarque : Il est possible de modifier très légèrement ces algorithmes de manière à obtenir aussi l'extension des concepts. Il suffit que les Mappers ajoutent l'information sur l'observation traitée, par exemple $(null, 2-bcd)$, $(null, 3-de)$ pour le premier. Puis le Reducer (ou Combiner) effectue un "ou" sur ces

[4] Si une ligne se répète, l'association est créée autant de fois. Cela ne change pas la suite du calcul.

observations ; par exemple, le Reducer effectuera un "ou" sur $(null, 2, 3-d)$ et $(null, 6-abcde)$ pour obtenir $(null, 2, 3, 6-d)$.

5.2.2 Algorithme des prédécesseurs

En améliorant l'algorithme de fermeture précédent, on peut obtenir tous les prédécesseurs immédiats (et d'autres) d'un concept donné, à l'aide d'une unique tâche MapReduce.

On se concentre sur les intensions. Rappelons tout d'abord que les concepts $C_1 = (Ext_1, Int_1)$ et $C_2 = (Ext_2, Int_2)$ sont ordonnés par :

$$C_1 \leq C_2 \iff Ext_1 \subseteq Ext_2 \iff Int_2 \subseteq Int_1$$

Ensuite, rappelons qu'étant donné un ensemble B d'attributs, ses prédécesseurs immédiats sont les fermés minimaux contenant strictement B. Ainsi, les prédécesseurs immédiats se trouvent parmi les ensembles fermés d'attributs de la forme $B \cup x$ où x est un attribut absent de B.

Donc, notre tâche MapReduce est la suivante :

- **Étape MAP** : Pour chaque observation g possédant strictement les attributs de B, i.e. $B \subset g'$, un ensemble de couples clé-valeur est généré. Ces couples clé-valeurs sont les (x, g') pour tous les $x \in g' \backslash B$.
- **Étape COMBINE-REDUCE** : Cette étape est la même que pour la fermeture, c'est à dire que pour chaque clé x la fermeture/"et binaire" est calculée.

La sortie produite par cette tâche MapReduce est l'ensemble des couples Clé-Valeur $(x, (B \cup x)'')$, où x est un attribut absent de B pour lequel il existe une observation contenant $B \cup x$. Cet ensemble d'intensions $(B \cup x)''$ contient évidemment les prédécesseurs immédiats de B mais aussi d'autres intensions ; les prédecesseurs immédiats étant les minimaux parmi ceux-ci.

Comme annoncé au début de ce chapitre, cet algorithme n'est pas optimal mais il est simple et passe à l'échelle.

Exemple 5.2.4. Considérons à nouveau l'exemple de la table 5.1 et utilisons notre algorithme pour le fermé d. Supposons à nouveau qu'il y a deux Mappers, recevant chacun trois lignes. Pour le premier Mapper (lignes 1 à 3), la ligne 1 est éliminée. La ligne 2 induit la création des couples clé-valeur suivant : (b, bcd) et (c, bcd). La ligne 3 induit la création du couple clé-valeur (e, de). Pour le second Mapper (lignes 4 à 6), seule la ligne 6 crée des couples clé-valeur, qui sont : $(a, abcde)$, $(b, abcde)$, $(c, abcde)$ et $(e, abcde)$. Vue la petitesse des données, les Combiners n'ont rien à faire. Il peut y avoir jusqu'à 4 Reducers selon la taille du cluster, ou, par exemple, deux Reducers traitant chacun 2 clés distinctes. Le reduceur traitant la clé a, retourne $(a, abcde)$, ce qui signifie que la fermeture de ad est $abcde$. Celui qui traite b, reçoit (b, bcd) et $(b, abcde)$, puis retourne (b, bcd). Cela signifie que la fermeture de bd est bcd. De même pour le traitement de la clé c, la fermeture de cd est bcd. Enfin, la clé e correspond aux deux couples

	a	b	c	d	e
1	x	x			
2		x	x	x	
3				x	x
4			x		
5	x		x		
6	x	x	x	x	x

TABLE 5.2 : Petit contexte utilisé pour illustrer le calcul d'implications utilisant la contraposée.

(e, de) et $(e, abcde)$, donc le Reducer retourne (e, de). Le résultat de l'algorithme est donc $(a, abcde)$, (b, bcd), (c, bcd) et (e, de). Les deux successeurs immédiats de d sont donc bcd et de.

5.3 Calculs des implications

Dans cette section, nous utiliserons à nouveau le contexte formel 5.1 donné une seconde fois dans la table 5.2.

5.3.1 Contraposée

L'implication $A \Longrightarrow B$ est équivalente à $\rceil B \Longrightarrow \rceil A$ et cette dernière est appelée contraposée de $A \Longrightarrow B$.

En termes d'attributs, la première forme affirme : toutes les observations qui contiennent les attributs A possèdent aussi les attributs B. La contraposée affirme : si une observation ne possède pas tous les attributs de B alors elle ne possède pas tous ceux de A non plus. Cette affirmation étant ambigue, elle peut être reformulée en : s'il existe un attribut de B qu'une observation n'a pas, alors il existe un attribut de A qu'elle ne possède pas non plus.

Exemple 5.3.1. Utilisons cela sur un exemple : prenons l'ensemble de tous les attributs $\{a, b, c, d, e\}$. Pour chaque attribut x, retenons les observations qui ne contiennent pas x.

On obtient :

- a : $2, 3, 4$.
- b : $3, 4, 5$.
- c : $1, 3$.
- d : $1, 4, 5$.
- e : $1, 2, 4, 5$.

	a	b	c	d	e
1			x	x	x
4	x	x		x	x
5		x		x	x

TABLE 5.3 : Sous-contexte complémentaire du contexte 5.2 obtenu en ne conservant que les observations vérifiant $\rceil d$.

	a	b	c	d	e
1			x	x	x
2	x				x
3	x	x	x		
4	x	x		x	x
5		x		x	x
6					

TABLE 5.4 : Sous-contexte renversé (i.e. complémentaire) du contexte 5.2.

En se concentrant sur l'attribut d, on dispose des lignes, qui ont la propriété "non d", à savoir 1, 4, 5. En renversant [5] le sous-contexte, on obtient celui de la table 5.3.

Finalement, en faisant un « et » binaire sur ces tuples, on obtient tous les attributs communs à ces lignes. Dans notre exemple, il s'agit de d, e ; en tenant compte du renversement de contexte, ces attributs sont caractérisés par : $\rceil d \rightarrow \rceil d \wedge \rceil e$. Maintenant, en revenant à l'implication d'origine, c'est à dire en prenant à nouveau la contraposée, on obtient $(e \vee d) \rightarrow d$. De cette implication, on ne retient que celle qui est informative : $e \rightarrow d$, qui est une implication de la base canonique directe. En traitant de même tous les attributs, nous obtenons donc les implications :

- $a \rightarrow a$
- $b \rightarrow b$
- $c \rightarrow c$
- $e \rightarrow d$
- $e \rightarrow e$

D'un point de vue global, le contexte renversé de celui de la table 5.2 est celui de la table 5.4

[5] Le contexte renversé est obtenu en prenant le complémentaire de la relation.

	a	b	c	d	e
1	x	x			
2		x	x	x	
6	x	x	x	x	x

TABLE 5.5 : Sous-contexte du contexte 5.2 contenant exactement les lignes possédant b.

	a	c	d	e
1	x			
2		x	x	
6	x	x	x	x

TABLE 5.6 : Sous-contexte du contexte 5.2 contenant exactement les lignes possédant b, sans l'attribut b.

L'usage de la contraposée est pertinent car il permet d'obtenir de manière contructive (sans éliminer des cas négatifs), ce genre d'implications.

Les calculs précédents nous permettent de trouver tous les générateurs minimaux n'ayant qu'un attribut dans leur prémisse.

5.3.2 Itération, approche incrémentale

Nous pouvons insérer cet usage de la contraposée dans un algorithme incrémental de calcul des générateurs minimaux de la manière suivante :

Exemple 5.3.2. A partir du contexte 5.2, on se restreint aux lignes ayant l'attribut b et l'on obtient le contexte de la table 5.5.

Cela revient à restreindre le treillis de concepts à l'idéal engendré par b. D'un point de vue de la logique, cela revient à donner à b le statut d'axiome : on suppose que b est vrai. Pour simplifier, on peut d'ailleurs complètement faire disparaître b. De sorte que nous avons maintenant le sous-contexte de la table 5.6.

Reproduisons maintenant le calcul précédent en sélectionnant les lignes ne possédant pas un attribut donné :

- $a : 2$
- $c : 1$
- $d : 1$
- $e : 1, 2.$

On prend les sous-contextes renversés :
Pour a, on obtient le sous-contexte de la table 5.7

	a	c	d	e
2	x			x

TABLE 5.7 : Sous-contexte complémentaire du contexte 5.6 contenant exactement les lignes ne possédant pas a.

	a	c	d	e
1		x	x	x

TABLE 5.8 : Sous-contexte complémentaire du contexte 5.6 contenant exactement les lignes ne possédant pas c.

Pour c, on obtient le sous-contexte de la table 5.8.
Pour d, on obtient le sous-contexte de la table 5.9.
Pour e, on obtient le sous-contexte de la table 5.10.
Enfin, comme précédemment, on fait un « et » binaire sur les tuples, puis on reprend la contraposée et l'on a :

- $e \vee a \rightarrow a$; on ne retient que : $e \rightarrow a$.
- $c \vee d \vee e \rightarrow c$; on retient $d \rightarrow c$ et $e \rightarrow c$.
- $e \rightarrow e$; il n'y a pas d'information.

Maintenant, n'oublions pas notre axiome ou hypothèse b et l'on conclut les implications suivantes :

- $b, e \rightarrow a$
- $b, d \rightarrow c$
- $b, e \rightarrow c$

En remarquant les prémisses communs, on obtient deux nouveaux générateurs minimaux :

- $b, e \rightarrow a, c$
- $b, d \rightarrow c$

	a	c	d	e
1		x	x	x

TABLE 5.9 : Sous-contexte complémentaire du contexte 5.6 contenant exactement les lignes ne possédant pas d.

	a	c	d	e
1		x	x	x
2	x			x

TABLE 5.10 : Sous-contexte complémentaire du contexte 5.6 contenant exactement les lignes ne possédant pas e.

Il s'agit de l'ensemble des générateurs minimaux de la forme $b*$ où $*$ est un attribut.

Cet exemple illustre une itération du procédé.

5.3.3 Conclusion

En parcourant tous les concepts du treillis, on obtient ainsi un algorithme incrémental, qu'il est également possible de déployer sur une architecture distribuée, permettant de calculer les générateurs minimaux.

Une approche plus raisonnable est certainement celle consistant à ne faire qu'une itération à partir d'un concept donné, à l'image du calcul présenté avec l'axiome b. En effet, ce calcul est basé sur un sous-contexte et est à la fois linéaire et distribué. Il permet d'obtenir une base locale d'implications.

L'intérêt de cette approche réside dans le fait qu'elle est constructive ; il ne s'agit plus de tester des cas et d'éliminer les mauvais, on fabrique les bons cas directement.

Enfin, si l'on cherche des implications du type $Set \rightarrow Attr$ avec $Attr$ imposé, on peut envisager de prendre le contexte renversé, sélectionner les lignes qui contiennent $Attr$ puis appliquer un OU. Alors, on obtient effectivement $\urcorner Attr \rightarrow \vee(\urcorner Set)$ et donc par contraposition, c'est bien ce qui était annoncé. Ces implications sont très proches de celle de la D-base [3, 2].

5.4 One Big Job

Dans cette section, nous continuons à améliorer notre algorithme MAP afin d'obtenir plus de résultats. Pour cela, remarquons qu'afin d'obtenir en une seule tâche MapReduce, une sortie de taille exponentielle, par exemple tous les concepts ou une base d'implications, nous avons besoin de générer une quantité exponentielle de couples Clé-Valeur. Ainsi, notre nouvel algorithme 8 est le suivant : pour chaque observation g, générer un couple Clé-Valeur (A, g') pour toute partie $A \subseteq g'$, soit $2^{|g'|}$ couples.

L'étape REDUCE est toujours la même, elle calcule la fermeture avec un « et » binaire. Ainsi, à la sortie de cette tâche MapReduce, on obtient un ensemble de couples Clé-Valeurs $A \rightarrow A''$ pour toute partie $A \subseteq M$ telle qu'il existe une observation g satisfaisant $A \subseteq g'$.

Pour être tout à fait rigoureux, il manquera la plupart du temps, le concept $\perp$, dès lors qu'il n'a pas d'observation.

Algorithme 8 : Algorithme MAP permettant d'obtenir tous les concepts et une base d'implications.

```
pour tous les o observation faire
    pour tous les A ⊆ o' faire
        // A parcourt l'ensemble des parties de l'ensemble o' Créer
           l'association clé-valeur (A, o');
```

	a l'attribut
Obs 1	1 2 3 4 5
Obs 2	1 3 5 6
Obs 3	1 2 3
Obs 4	1 2 4
Obs 5	2 4

TABLE 5.11 : Un contexte avec 5 observations et 6 attributs.

Donnons un exemple du déroulement de cet algorithme.

Exemple 5.4.1. Nous changeons la manière de décrire le contexte initial, cela rendra les calculs plus faciles à lire. Ainsi, le contexte initial est donné dans la table 5.11, où les attributs sont numérotés de 1 à 6.

Quand on traite l'observation 1, $32 = 2^5$ couples de clé-valeur sont générés. En effet, l'observation $o = 1$ possède 5 attributs (numérotés de 1 à 5). L'ensemble des parties de l'ensemble $o' = \{1, 2, 3, 4, 5\}$ possède $2^5 = 32$ éléments. Pour chacune de ces 32 parties $A \subseteq o'$, un couple clé-valeur (A, o') est créé. De même, quand on traite l'observation 2, 16 couples de clé-valeur sont générés car l'ensemble des attributs de l'observation 2 est $\{1, 3, 5, 6\}$ et que cet ensemble possède $2^4 = 16$ parties. Enfin, quand on traite les observations 3 et 4, 8 couples de clé-valeur sont générés. La table 5.12 donne ces couples pour l'observation 4. Quand on traite l'observation 5, 4 couples de clé-valeur sont générés.

Considérons maintenant le Reducer qui sera en charge de la clé $\{3, 5\}$. Ce Reducer recevra les couples clé-valeur de la table 5.13.

Comme ce Reducer calcule un « et » booléen, il retournera le couple :

$$(\{3, 5\}, \{1, 3, 5\})$$

La sortie complète de cette tâche MapReduce est donnée dans la table 5.14.

Mis à part le concept $\perp$ qui n'a pas d'observation, nous obtenons tous les concepts. De plus, pour une valeur donnée, i.e. un fermé, en considérant ses clés minimales, on obtient ses générateurs minimaux.

Clé	Valeur
$\{1,2,4\}$	$1,2,4$
$\{1,2\}$	$1,2,4$
$\{1,4\}$	$1,2,4$
$\{2,4\}$	$1,2,4$
$\{1\}$	$1,2,4$
$\{2\}$	$1,2,4$
$\{4\}$	$1,2,4$
$\emptyset$	$1,2,4$

TABLE 5.12 : Couples de clé-valeur générés par l'observation 4.

Clé	Valeur
$\{3,5\}$	$1,2,3,4,5$
$\{3,5\}$	$1,3,5,6$

TABLE 5.13 : Couples clé-valeur reçus par le Reducer traitant la clé $\{3,5\}$.

Clé	Valeur	Clé	Valeur
$\{1,2,3,4,5\}$	$\{1,2,3,4,5\}$	$\{1,2,3,4\}$	$\{1,2,3,4,5\}$
$\{1,2,3,5\}$	$\{1,2,3,4,5\}$	$\{1,2,4,5\}$	$\{1,2,3,4,5\}$
$\{1,3,4,5\}$	$\{1,2,3,4,5\}$	$\{2,3,4,5\}$	$\{1,2,3,4,5\}$
$\{1,2,5\}$	$\{1,2,3,4,5\}$	$\{1,3,4\}$	$\{1,2,3,4,5\}$
$\{1,4,5\}$	$\{1,2,3,4,5\}$	$\{2,3,4\}$	$\{1,2,3,4,5\}$
$\{2,3,5\}$	$\{1,2,3,4,5\}$	$\{2,4,5\}$	$\{1,2,3,4,5\}$
$\{3,4,5\}$	$\{1,2,3,4,5\}$	$\{2,5\}$	$\{1,2,3,4,5\}$
$\{3,4\}$	$\{1,2,3,4,5\}$	$\{4,5\}$	$\{1,2,3,4,5\}$
$\{1,3,5,6\}$	$\{1,3,5,6\}$	$\{1,3,6\}$	$\{1,3,5,6\}$
$\{1,5,6\}$	$\{1,3,5,6\}$	$\{3,5,6\}$	$\{1,3,5,6\}$
$\{1,6\}$	$\{1,3,5,6\}$	$\{3,6\}$	$\{1,3,5,6\}$
$\{5,6\}$	$\{1,3,5,6\}$	$\{6\}$	$\{1,3,5,6\}$
$\{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$	$\{1,2,3\}$
$\{1,2,4\}$	$\{1,2,4\}$	$\{1,4\}$	$\{1,2,4\}$
$\{1,3,5\}$	$\{1,3,5\}$	$\{1,5\}$	$\{1,3,5\}$
$\{3,5\}$	$\{1,3,5\}$	$\{5\}$	$\{1,3,5\}$
$\{1,2\}$	$\{1,2\}$	$\{1,3\}$	$\{1,3\}$
$\{3\}$	$\{1,3\}$	$\{2,4\}$	$\{2,4\}$
$\{4\}$	$\{2,4\}$	$\{1\}$	$\{1\}$
$\{2\}$	$\{2\}$	$\{\}$	$\{\}$

TABLE 5.14 : Sortie complète de la tâche MapReduce.

5.5 Conclusion

Ce chapitre repose sur deux choix : celui du paradigme MapReduce et celui d'une implémentation particulière Hadoop.

Il possible de conserver l'approche MapReduce et de changer d'implémentation. Cela a déjà été fait [190] avec Twister qui est plus adapté à une approche itérative que Hadoop. Le but à atteindre est alors de trouver l'implémentation la plus adaptée à la FCA, en faisant une étude comparative et ultimement de produire une implémentation prenant en compte dès le début les spécificités de la FCA.

Une approche spécifique à la FCA et aux pattern structures a également été explorée [115].

En conservant l'approche « calcul distribué » (Scale Out), il est possible de changer de paradigme et par exemple utiliser MPI (Message Passing Interface) plutôt que MapReduce. Là encore, une comparaison de ces paradigmes dans le cadre de la FCA serait utile.

Enfin, il faudrait mener une série d'expérimentations avec des données massives et exploitant une architecture distribuée et comparer avec des algorithmes standards afin de valider cette approche.

Chapitre 6

Illustrations et expérimentations

Sommaire

6.1 Exemple d'utilisation de LattExp

6.1.1 Présentation du jeu de données utilisé

Pour cette expérience, nous avons utilisé le contexte donné dans la table 6.1. Dans ce contexte, des femmes de Caroline du Sud (les observations) ont participé à des évènements (attributs).

Ce contexte est extrait du jeu de données obtenu à partir de l'étude sur les femmes de Caroline du Sud [75, 74]. Le jeu de données original est donné en annexe 7.5.7. Le treillis de concepts des données d'origine possède 65 noeuds. Nous avons extrait ce sous-ensemble (voir table 6.1) pour avoir moins de concepts et de meilleures visualisations. Les choix effectués pour sélectionner ce

	1	2	3	4	5	6	7	8	9
Evelyn	x	x	x	x	x	x		x	x
Laura	x	x	x		x	x	x	x	
Theresa		x	x	x	x	x	x	x	x
Brenda	x		x	x	x	x	x	x	
Charlotte			x	x	x		x		
Frances			x		x	x		x	
Eleanor					x	x	x	x	
Katherine								x	x
Sylvia							x	x	x
Nora						x	x		x
Helen							x	x	

TABLE 6.1 : Jeu de données extrait de l'étude sur les femmes de Caroline du Sud.

sous-contexte utilisent les résultats de M.G. Everett and S.P. Borgatti [67]. Avec ce choix, nous nous attendons à obtenir deux groupes de femmes.

A partir de ces données, nous avons obtenu un treillis possédant 36 concepts. Ce dernier est donné dans les figures 6.1 et 6.2.

Remarquons que les visualisations produites par ConExp ont été améliorées autant que possible alors que celle produites par LattExp, non.

A partir de ces représentations et avec un peu de temps [1], nous pouvons remarquer que :

- Un groupe/communauté est constitué de Brenda, Laura, Charlotte, Eleanor, Frances et Helen. Brenda et Laura constituent le noyau de ce groupe, alors que les autres sont à la périphérie.
- Un plus petit second groupe est constitué de Sylvia, Katherine et Helen. Sylvia est le noyau de ce groupe.
- Remarquons qu'Helen est dans les deux groupes.
- Theresa et Evelyn font le lien/pont entre les groupes.
- Enfin, Nora est isolée.

6.1.2 Le treillis des tolérances

Afin de visualiser l'espace dans lequel la navigation va se faire, nous avons calculé le treillis des tolérances du treillis initial. Remarquons cependant que ce calcul n'est pas nécessaire. Il est possible de naviguer dans ce treillis sans le calculer. Ce treillis des tolérances de notre jeu de données est donné dans la figure 6.3.

[1] En utilisant le logiciel ConExp, de manière tout à fait empirique, en regroupant progressivement les concepts, j'ai fini par organiser les données de sorte que les remarques faites ont été possible.

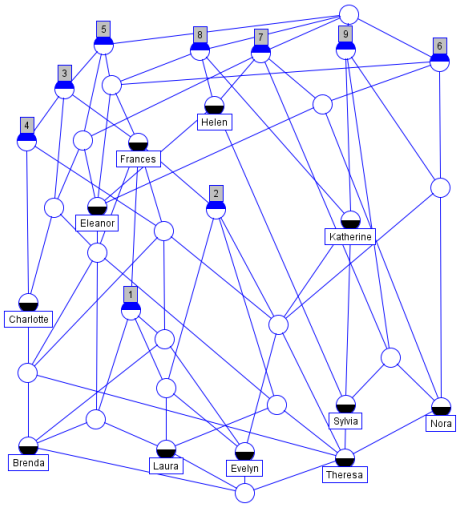


FIGURE 6.1 : Le treillis de concepts déduit du contexte 6.1, en utilisant ConExp [192].

Le treillis initial (figure 6.1) correpond au noeud $\top$ du treillis de la figure 6.3. Ce sera le point de départ de notre navigation. Donnons avant une petite explication sur les noms utilisés pour les noeuds irréductibles de la figure 6.3. Prenons par exemple le noeud "Eleanor-3". Cette relation de tolérance est obtenue en prenant le contexte initial, en ajoutant une croix dans la cellule $(Eleanor, 3)$ puis en calculant la plus petite relation de tolérance contenant ce contexte.

Remarquons que parmi ces relations de tolérances, certaines sont des congruences. Pour illustrer ce point, le treillis des congruences a été calculé et est donné dans la figure 6.4

6.1.3 Navigation

Comme nous l'avons déjà constaté, notre treillis initial est un peu trop grand pour être analysé facilement ; la représentation est difficilement lisible, quelque soit l'objectif de l'utilisateur. Donc, nous commençons à réduire sa taille en calculant le treillis quotient de la tolérance "Frances-1". Nous obtenons alors le treillis donné dans les figures 6.5 et 6.6. Suivant ce que l'on cherche dans le treillis, cette réduction peut ou non être pertinente.

Ce treillis contient toujours 28 noeuds. Donc, il est à peine plus facile à lire que le précédent. Remarquons cependant que son analyse conduit aux mêmes conclusions que le treillis initial. Le regroupement de certains concepts n'a pas

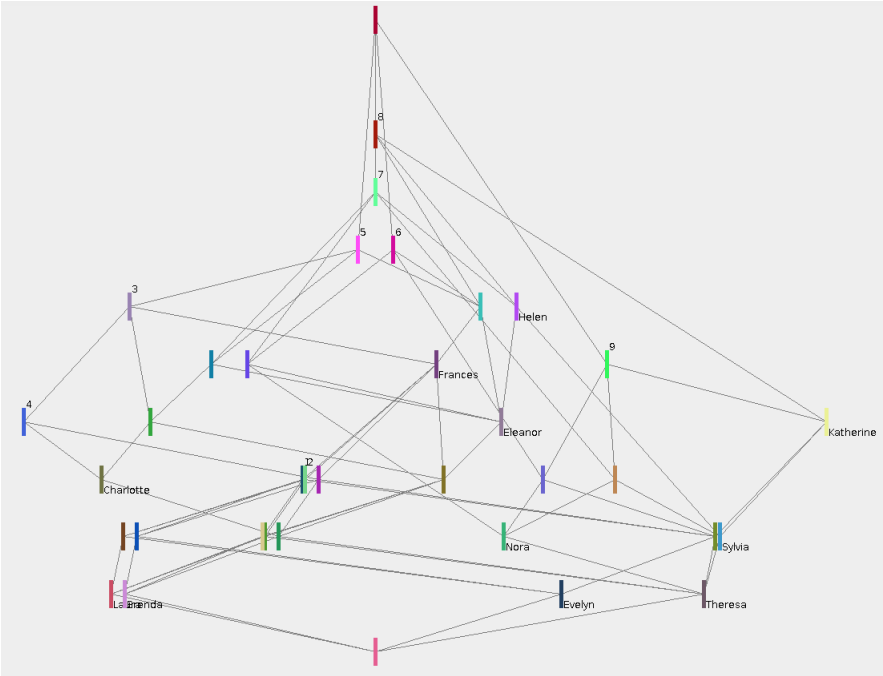


FIGURE 6.2 : Le treillis de concepts déduit du contexte 6.1, en utilisant LattExp. La représentation n'est pas meilleure qu'avec ConExp.

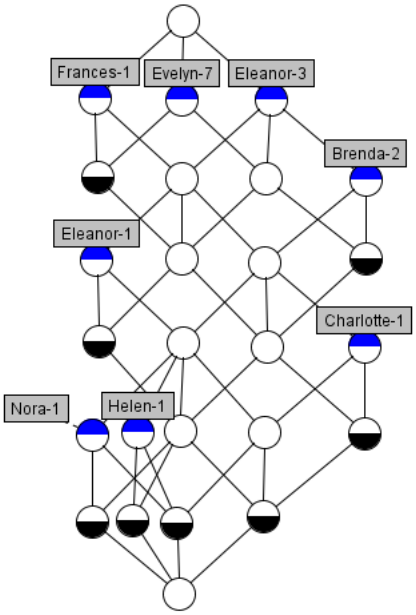


FIGURE 6.3 : Le treillis des tolérances du jeu de données 6.1.

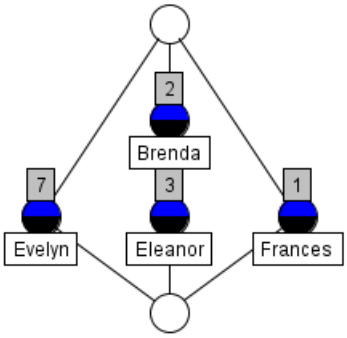


FIGURE 6.4 : Treillis des congruences du jeu de données 6.1.

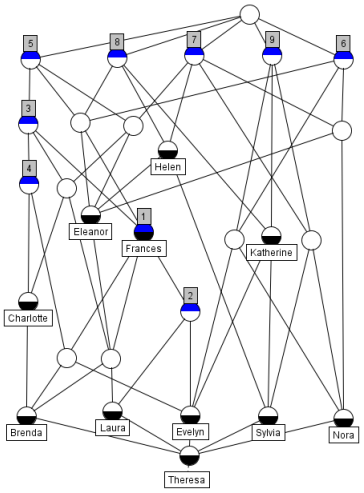


FIGURE 6.5 : Treillis quotient de la tolerance "Frances-1" produit par ConExp.

changé la sémantique. En particulier, il est encore plus clair maintenant que Theresa a une position de lien et comme elle est tout en bas du treillis, cela révèle qu'elle a participé à beaucoup d'évènements.

La figure 6.6 produite par LattExp contient toujours tous les noeuds du treillis intial mais certains d'entre eux ont été groupés en accord avec les blocs de la relation de tolérance. La méthode de visualisation prend en compte la relation de tolérance. Les noeuds d'un même bloc sont de la même couleur, et les arêtes entre deux noeuds d'un même bloc sont de la couleur commune des noeuds. Lorsque deux noeuds sont dans des blocs différents, l'arête est en gris.

Pour avancer dans notre navigation, nous avons calculé la tolérance engendrée par "Frances-1", "Eleanor-3" et "Evelyn-7". Le treillis quotient de cette relation est donné sur les figures 6.7 et 6.8.

Maintenant, avec un treillis contenant uniquement treize noeuds, les conclusions sont plus faciles à tirer et restent les mêmes que dans le treillis intial. Remarquons que dans la figure 6.8 produite par LattExp, nous avons tous les noeuds du treillis initial ; cependant, ils ont été groupés en accord avec les blocs de la tolérance. De plus, cette tolérance n'est pas une congruence car certains concepts sont dans plusieurs blocs. Ces concepts sont représentés plusieurs fois et sont reliés par une arête en rouge.

Si nous continuons à descendre dans notre navigation, la plupart des treillis quotient n'ont plus que quatre noeuds. Cette réduction de taille est excessive et les données ne peuvent plus être interprétées. Cependant, notons en particulier que c'est le cas de "Charlotte-1", "Nora-1" et "Helen-1". Cela révèle leur position

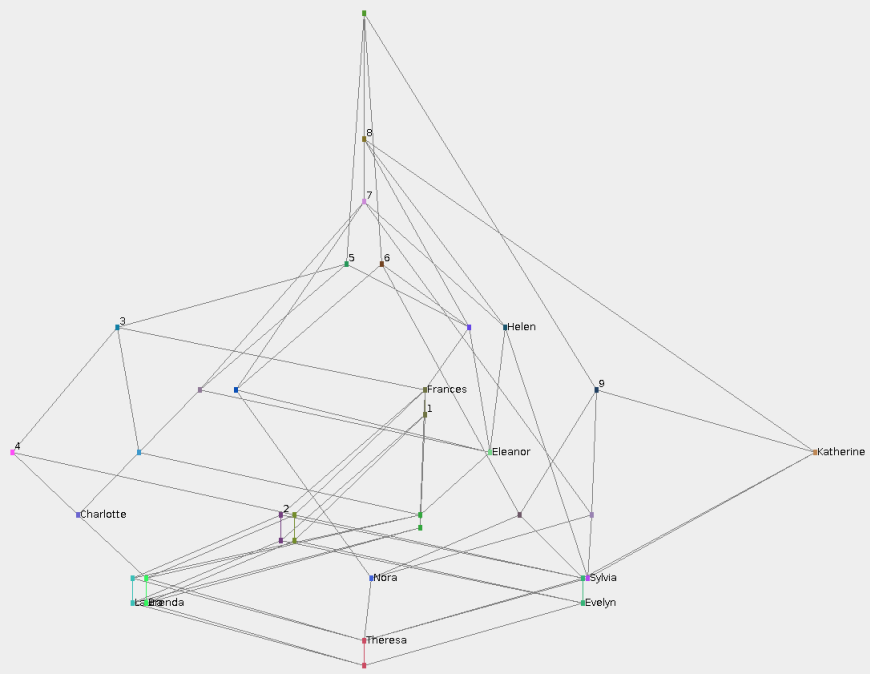


FIGURE 6.6 : Treillis quotient de la tolerance "Frances-1" produit par LattExp. Dans cette représentation, les noeuds d'un même blocs sont seulement rapprochés pas amalgamés ; de plus les noeuds présents dans plusieurs blocs sont représentés plusieurs fois.

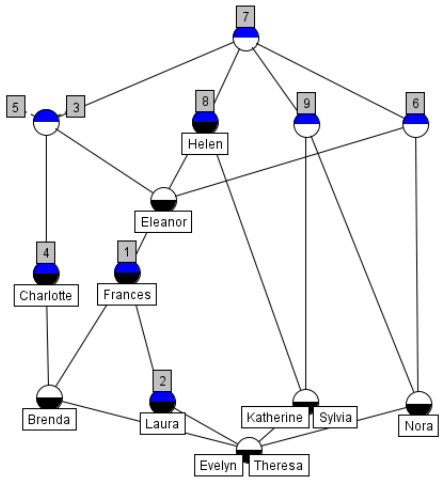


FIGURE 6.7 : Le treillis facteur de la tolérance engendrée par "Frances-1", "Eleanor-3" et "Evelyn-7" par ConExp.

particulière. Nous avions déjà remarqué la position isolée de Nora. Par ailleurs, nous avions vue Helen comme membre des deux groupes ; avec cette transformation elle a maintenant plutôt un rôle de pont entre les groupes, ce qui revient pratiquement au même. Nous découvrons maintenant que même si Charlotte est dans le premier groupe, elle y a une position un peu isolée. Par ailleurs, ces tolérances sont également intéressantes pour leurs blocs.

6.1.4 Blocs et implications

Considérons maintenant la relation de tolérance engendrée par "Charlotte-1". Son treillis quotient ne possède que 4 noeuds et donc n'a pas d'intérêt. Cependant, les blocs de cette relation sont eux particulièrement intéressant. Rappelons que les blocs d'une relation de tolérance sont des intervalles. Pour la tolérance que nous considérons, les quatre blocs sont : $[\bot, *]$, $[Evelyn, *]$, $[*, 7]$, $[*, \top]$, où $*$ désigne un concept qui n'a rien de particulier. Ces quatre blocs sont donnés dans les figures 6.9, 6.10, 6.11, 6.12.

Nous remarquons évidemment que ces blocs vont par paires : *Evelyn* avec $\bot$, et 7 avec $\top$. Ces sous-treillis isomorphes ne sont pas faciles à repérer à l'intérieur du treillis initial. Il s'agit cependant d'un cas particulier pour cette relation. On peut sans doute interpréter cela à cause de la place particulière d'Evelyn, qui est à la périphérie. Et l'on peut tirer la même conclusion pour l'évènement 7.

Le point le plus intéressant pour ces blocs concerne la base canonique (di-

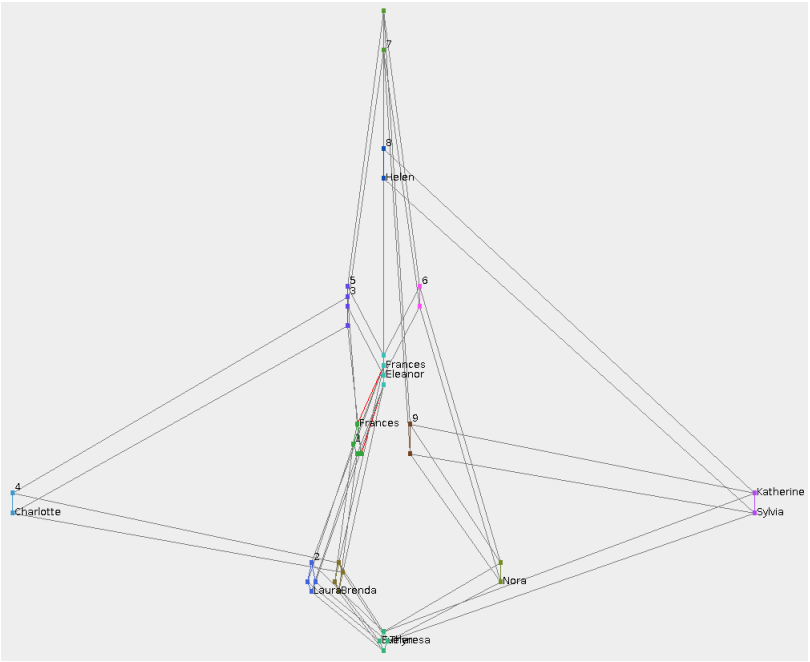


FIGURE 6.8 : Le treillis facteur de la tolérance engendrée par "Frances-1", "Eleanor-3" et "Evelyn-7" par LattExp. Dans cette représentation, les noeuds d'un même blocs sont seulement rapprochés pas amalgamés ; de plus les noeuds présents dans plusieurs blocs sont représentés plusieurs fois.

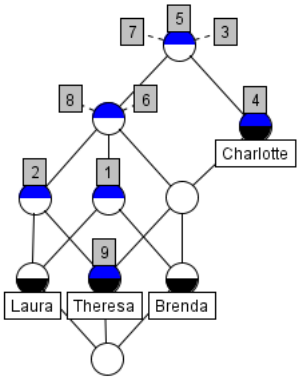


FIGURE 6.9 : Bloc $\perp$.

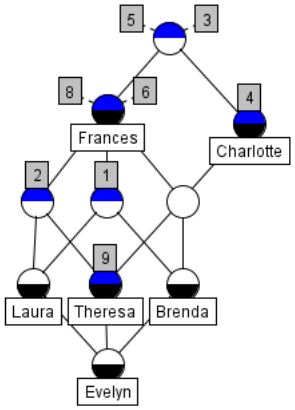


FIGURE 6.10 : Bloc *Evelyn*.

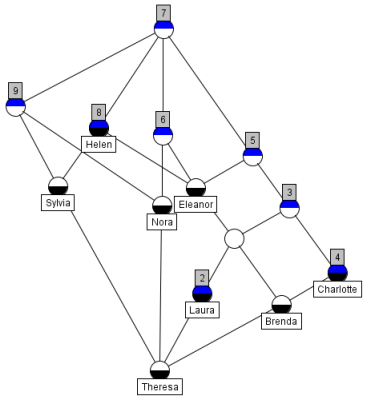


FIGURE 6.11 : Bloc 7.

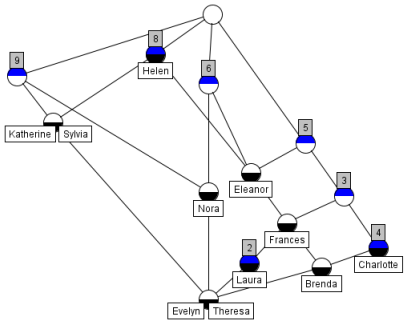


FIGURE 6.12 : Bloc $\top$.

$1 \rightarrow 3\ 5\ 6\ 8$
$2 \rightarrow 3\ 5\ 6\ 8$
$3 \rightarrow 5$
$4 \rightarrow 3\ 5$
$5\ 6 \rightarrow 8$
$5\ 8 \rightarrow 6$
$5\ 9 \rightarrow 2\ 3\ 4\ 6\ 8$
$6\ 8 \rightarrow 5$
$2\ 3\ 4\ 5\ 6\ 8 \rightarrow 9$

TABLE 6.2 : Base canonique (ou de Guigues et Duquenne) du treillis de la figure 6.1

recte). Dans la suite, nous nous concentrons sur la base canonique (ou base de Guigues et Duquenne) mais les mêmes remarques peuvent être faites dans le cas de la base canonique directe.

La base canonique du treillis initial est donnée dans la table 6.2.

Dans la table 6.3, la base canonique des quatres blocs est donnée.

Mises à part les implications avec des prémisses vides, les blocs isomorphes ont la même base. C'est évidemment vrai dans le cas général puisque cette base est "canonique" au sens où elle ne dépend que du treillis. De plus, comme dans cet exemple les blocs sont "suffisamment" grands, leur base respective permet de reconstruire la base du treillis d'origine. Cette question théorique est actuellement en cours d'étude.

Base canonique du bloc $\perp$	Base canonique du bloc *Evelyn*
$\emptyset \rightarrow 3\ 5\ 7$	$\emptyset \rightarrow 3\ 5$
$1\ 3\ 5\ 7 \rightarrow 6\ 8$	$1\ 3\ 5 \rightarrow 6\ 8$
$2\ 3\ 5\ 7 \rightarrow 6\ 8$	$2\ 3\ 5 \rightarrow 6\ 8$
$3\ 5\ 6\ 7 \rightarrow 8$	$3\ 5\ 6 \rightarrow 8$
$3\ 5\ 7\ 8 \rightarrow 6$	$3\ 5\ 8 \rightarrow 6$
$3\ 5\ 7\ 9 \rightarrow 2\ 4\ 6\ 8$	$3\ 5\ 9 \rightarrow 2\ 4\ 6\ 8$
$2\ 3\ 4\ 5\ 6\ 7\ 8 \rightarrow 9$	$2\ 3\ 4\ 5\ 6\ 8 \rightarrow 9$
Base canonique du bloc $\top$	Base canonique du bloc 7
	$\emptyset \rightarrow 7$
$2 \rightarrow 3\ 5\ 6\ 8$	$2\ 7 \rightarrow 3\ 5\ 6\ 8$
$3 \rightarrow 5$	$3\ 7 \rightarrow 5$
$4 \rightarrow 3\ 5$	$4\ 7 \rightarrow 3\ 5$
$5\ 6 \rightarrow 8$	$5\ 6\ 7 \rightarrow 8$
$5\ 8 \rightarrow 6$	$5\ 7\ 8 \rightarrow 6$
$5\ 9 \rightarrow 2\ 3\ 4\ 6\ 8$	$5\ 7\ 9 \rightarrow 2\ 3\ 4\ 6\ 8$
$6\ 8 \rightarrow 5$	$6\ 7\ 8 \rightarrow 5$
$2\ 3\ 4\ 5\ 6\ 8 \rightarrow 9$	$2\ 3\ 4\ 5\ 6\ 7\ 8 \rightarrow 9$

TABLE 6.3 : Bases canoniques (ou de Guigues et Duquenne) des quatre blocs de la relation de tolérance "Charlotte-1".

6.1.5 Métriques

Donnons quelques détails supplémentaires à propos des métriques dans LattExp. Supposons que l'utilisateur soit à l'étape de navigation faisant intervenir le treillis à 13 noeuds. Alors, il apprend qu'il peut continuer vers le bas (treillis plus petits) de 5 façons différentes, mais 4 d'entre elles donnent un treillis à 2 noeuds. La dernière donne un treillis dont la taille est 92% de la taille du treillis initial (soit 12 noeuds).

L'utilisateur peut également remonter (treillis plus grands) de 4 manières. Les treillis ont alors une taille respective de 277%, 200%, 123%, 108%.

LattExp donne aussi plus de détails à propos de ces treillis. Il indique les ratios des dimensions des contextes réduits. Par exemple, pour le treillis d'une taille de 92%, le ratio des nombres d'attributs est 85% et le ratio des nombres d'observations est aussi 85%. Ainsi, le treillis et le contexte sont tous les deux un peu plus petit.

Enfin, LattExp indique comment le contexte a été modifié. En prenant, par exemple un des treillis de taille 2, alors on obtient les informations de la table 6.4. Les ratios correspondent aux quantités de cellules vides qui ont été remplies.

6.1.6 Conclusion

Notre navigation nous a conduit d'un treillis initial "grand" à un plus petit et plus simple. Cependant, il est également possible de commencer avec une vue

Ligne	Ratio	Colonne	Ratio
Brenda	100.0%	1	0.0%
Charlotte	80.0%	2	100.0%
Eleanor	75.0%	3	100.0%
Evelyn	NaN%	4	100.0%
Frances	100.0%	5	100.0%
Helen	85.7%	6	100.0%
Katherine	83.3%	7	NaN%
Laura	100.0%	8	100.0%
Nora	83.3%	9	100.0%
Sylvia	83.3%		
Theresa	NaN%		

TABLE 6.4 : Ratio de cellules remplies par ligne/colonne. "NaN" indique que la ligne/colonne était déjà pleine.

réduite, par exemple un treillis quotient possédant seulement quatre noeuds, puis de remonter en développant ces noeuds. Plus précisément, à chaque étape de la navigation, il est possible de choisir entre réduire et agrandir le treillis actuel.

Remarquons que le choix du point de départ de la navigation est problématique. D'une part, si l'on commence avec le treillis le plus grand, beaucoup de calculs peuvent être faits pour rien et la visualisation peut être illisible. D'autre part, démarrer tout en bas, avec un treillis quotient d'un seul noeud, peut amener à une navigation longue et fastidieuse. Une question actuellement à l'étude est de trouver un moyen de démarrer la navigation au « beau milieu [2] » de l'espace ou mieux encore que la première représentation soit le résultat d'une requête utilisateur.

Le squelette d'un treillis [88] est une relation de tolérance particulière qui semble présenter un bon compromis pour démarrer la navigation. En effet, dans l'exemple que nous avons considéré, le squelette ne possède que deux noeuds, mais parmi ses successeurs immédiats se trouve le treillis à 13 noeuds qui nous intéressait.

En ce qui concerne cette expérience en particulier, ce jeu de données extrait devrait également être étudié avec des outils classiques, comme UCINET [3], dans le but de comparer les résultats au niveau de la constitution des communautés.

[2] Cette notion reste à définir scientifiquement.
[3] https://sites.google.com/site/ucinetsoftware/home (lien vérifié le 26 septembre 2017)

Facteurs	Densité=20%	Densité=50%	Densité=80%
1	87, 40%	70, 50%	1, 50%
2	9, 70%	21, 40%	9, 90%
3	2, 60%	6, 20%	18, 70%
4	0, 30%	0, 50%	23, 40%
5		1, 40%	28, 50%
6			18, 00%

TABLE 6.5 : Proportion du nombre de facteurs de la décomposition sous-directe des contextes en fonction de leur densité.

6.2 Expériences sur les relations de congruence

6.2.1 Décomposition sous-directe

Dans cette section, nous avons mené une étude empirique dans le but de mieux comprendre l'impact de la décomposition sous-directe sur les contextes, en utilisant différentes densités (rapport du nombre de croix sur le nombre de cases). Tous les tests ont été réalisés à l'aide du projet *TheGalactic* disponible à l'adresse :

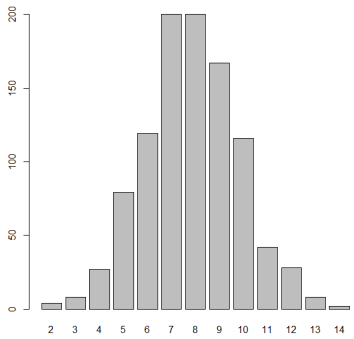
http ://thegalactic.github.io/java-lattices/

Un millier de contextes possédant 10 observations et 5 attributs ont été générés aléatoirement. Cette expérience a été réalisée trois fois, avec trois densités différentes de 20%, 50% et 80%. La figure 6.13 présente le nombre de treillis obtenus (axe des ordonnées) en fonction de leur taille (en nombre de concepts) pour chacune des trois densités. Nous pouvons observer deux Gaussiennes pour les deux premiers cas, alors que le dernier cas possède un comportement erratique. Cependant, nous pouvons observer que grosso-modo, plus la densité est élevée, plus les treillis sont grands. Il semble donc que la densité aura un impact sur le processus de décomposition comme on peut le prévoir intuitivement.

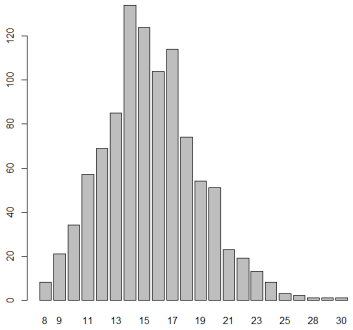
Le nombre de facteurs des décompositions est donné dans la table 6.5. On peut observer que ce nombre croît avec la densité du contexte initial puisque les treillis correspondant ont plus de noeuds. Lorsque la densité du contexte est de 20%, 87.40% des contextes sont irréductibles. Mais lorsque la densité est de 80%, seulement 1.5% des contextes sont irréductibles et 70% d'entre eux ont une décomposition faisant intervenir au moins 4 facteurs. Ainsi, les treillis semblent plus se décomposer lorsque leur contexte est plus dense (le nombre de facteurs est plus important et le nombre d'irréductibles est plus faible).

Observons maintenant la taille des facteurs avec deux valeurs de densité différentes, 20% et 50%. La figure 6.14 donne le nombre de treillis en fonction de leur nombre. On observe évidemment que les treillis plus petits donnent des facteurs plus petits. On peut également observer que, pour ces deux densités, le nombre de facteurs le plus courant est 2.

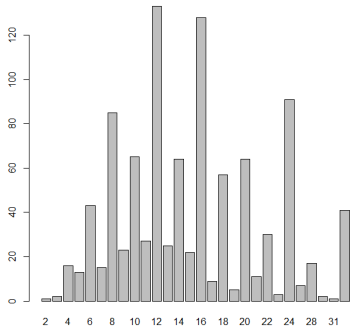
Pour les derniers tests, nous utilisons des contextes dont la densité est de 80% et nous calculons le rapport entre le nombre de noeuds (resp. arêtes) du



(a) Densité de 20%

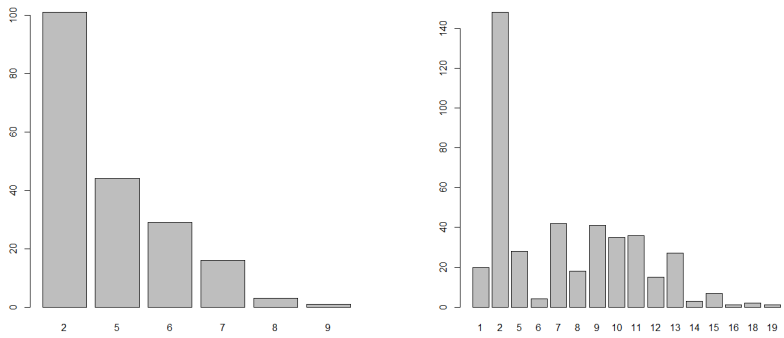


(b) Densité de 50%



(c) Densité de 80%

FIGURE 6.13 : Nombre de trellis en fonction de leur taille (en nombre de concepts).



(a) Densité 20% (b) Densité 50%

FIGURE 6.14 : Nombre de treillis en fonction du nombre de facteurs dans leur décomposition.

Facteurs	% de noeuds	% d'arêtes
1	100%	100%
2	97,90%	97,20%
3	89,20%	84,60%
4	85,00%	77,40%
5	80,60%	71,80%
6	68,00%	57,30%

TABLE 6.6 : Proportion de noeuds et arêtes vides, c'est à dire inutiles dans les treillis produit.

treillis initial et du treillis produit. Nous obtenons alors la table 6.6 qui montre que plus la décomposition a de facteurs plus le treillis produit est grand et donc que la quantité de noeuds inutiles (vides) dans le produit est importante ; cela montre que plus la décomposition est bonne, moins il faut cherche à incarner le treillis initial dans le treillis produit.

Nous avons également conduit une expérimentation avec des contextes contenant 40 observations et 15 attributs. Une centaine de contextes ont été générés pour deux densités 20% et 50%. Tous étaient irréductibles. Donc, cette approche de réduction via les relations de congruences est trop restrictive et doit être assouplie, par exemple en utilisant des relations de tolérance.

6.2.2 Doublement de convexe

Pour illustrer la construction de doublement inversée, nous utiliserons le treillis de la figure 6.15 et son contexte réduit donné dans la table 6.7 qui a déjà été

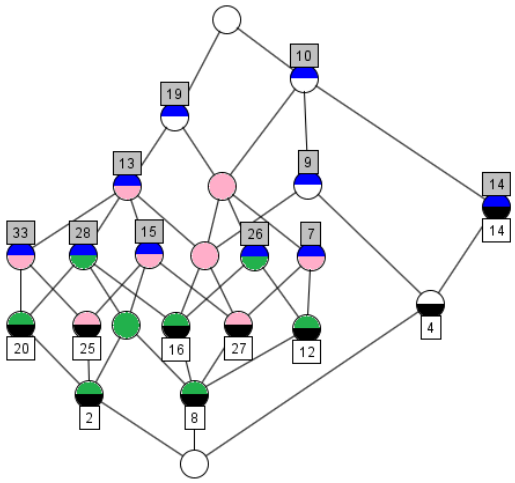


FIGURE 6.15 : Treillis de la figure 6.17 dont le convexe a été doublé. Les noeuds de chaque convexe sont en rose et vert.

complétée avec les flèches. Les noeuds en rose et vert correspondent aux deux occurrences du convexe.

A partir du treillis de la figure 6.15, on calcule la table 6.8 des sous-contextes monogènes et ensuite on obtient le treillis des relations de congruence donné à la figure 6.16. La table 6.8 a été réduite après les calculs. Dans ce cas particulier, il y a deux atomes.

Comme les deux atomes de la figure 6.16 satisfont la condition de Geyer, le treillis de la figure 6.15 admet deux décompositions.

En utilisant un seul de ces deux atomes, à savoir celui contenant l'observation 25, nous obtenons le sous-contexte flèche-fermé de la table 6.9, qui satisfait à la condition de Geyer. Alors son treillis de concepts, qui est aussi le treillis quotient de la relation de congruence choisie, peut être calculé et visualisé sur la figure 6.17.

Sur la figure 6.17, les noeuds du convexe C sont en rouge, et lorsque la construction de doublement est appliquée à ce treillis, avec cet ensemble convexe, le treillis de la figure 6.15 est obtenu. Les noeuds en rose et en vert correspondent aux deux occurrences du convexe doublé. Ce treillis est obtenu à partir du contexte réduit donné dans la table 6.9, qui est un des deux atomes du treillis de la figure 6.16.

Afin de confronter notre décomposition vis à vis de sa capacité à réduire la taille du treillis, nous avons mené une première série d'expériences.

Un lot de 500 contextes aléatoires ayant 20 observations, 10 attributs et une densité de 75% a été généré. Seulement 42 d'entre eux (8.4%) avaient une décomposition non triviale. Cependant, 17 de ces contextes (40%) ont conduit

	9	10	14	19	33	15	28	7	26	13
2	↓	↕	↓	×	×	×	×	↓	↓	×
4	×	×	×	↕	↓	↓	↓	↓	↓	↓
8	×	×	↕	×	↕	×	×	×	×	×
12	↕	×	↑	×	○	↓	↓	×	×	↕
14	↕	×	×	↑	○	○	○	○	○	○
25	○	↑	○	×	×	×	↕	○	○	×
20	○	↑	○	×	×	↕	×	○	○	×
27	×	×	↑	×	↑	×	↕	×	↕	×
16	×	×	↑	×	↑	↕	×	↕	×	×

TABLE 6.7 : Le contexte réduit du treillis de la figure 6.15 complété avec les flèches et ○ (voir 2.3.6).

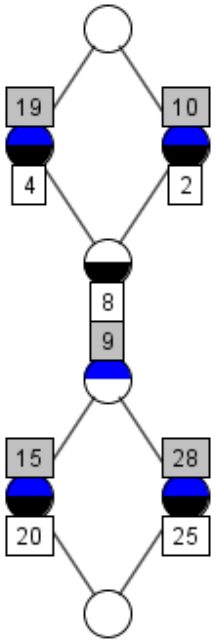


FIGURE 6.16 : Le treillis des relations de congruence du treillis de la figure 6.15

	9	10	19	15	28
2		×			
4			×		
8		×	×		
25	×	×	×		×
20	×	×	×	×	

TABLE 6.8 : Le contexte *réduit* calculé pour obtenir le treillis des relations de congruence (figure 6.16).

	10	13	14	19	26	28	33	9
12	×			×	×			
14	×		×					
2		×		×		×	×	
25		×		×			×	
27	×	×		×				×
4	×		×					×
8	×			×	×	×		×

TABLE 6.9 : Le contexte réduit du treillis quotient de la figure 6.17

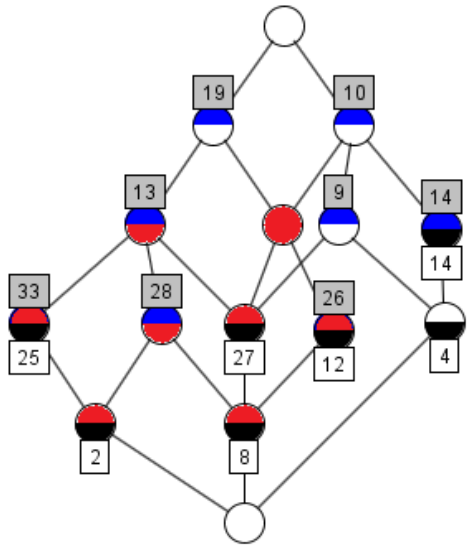


FIGURE 6.17 : Un treilis où les concepts-objets $(\{o\}'', \{o\}')$ ont un nombre au-dessous et les concepts-attributs $(\{a\}', \{a\}'')$ ont un nombre au-dessus.

Ratio des nombres de noeuds	Nombres et proportions de contextes
50%	17 - 40%
50% to 60%	9 - 21%
60% to 70%	6 - 14%
70% to 80%	5 - 12%
80% to 90%	4 - 10%
90% to 100%	1 - 3%

TABLE 6.10 : Taux de réduction de la taille des treillis et proportions de treillis réduits.

à un treillis réduit à un taux de 50% alors que le reste (60%) de ces contextes ont conduit à des treillis ayant une taille réduite de plus de 50%. La table 6.10 donne ces ratios où la première colonne représente la taille de la réduction (*i.e.*, le nombre de noeuds du treillis réduit sur le nombre de noeuds du treillis initial) et la seconde colonne indique le nombre et la proportion de contextes pour lesquels le taux de rédution est dans l'intervalle donné en première colonne. Nous pouvons conclure que la décomposition est rarement possible, mais lorsqu'elle l'est, elle conduit souvent à une réduction significative de la taille du treillis.

Tous ces calculs ont été menés avec le projet TheGalactic disponible à l'adresse :

http ://thegalactic.github.io/

et les treillis ont été représentés à l'aide de ConExp disponible à l'adresse :

http ://conexp.sourceforge.net/

6.3 Expériences sur les relations de tolérance

Dans cette section, nous avons mené une série d'expériences sur les relations de tolérance. Avec ces expériences, nous souhaitions observer si, en général, il y a beaucoup ou peu de relations de tolérance pour un treillis donné.

Nous avons généré aléatoirement des contextes $K = (O, A, R)$. Ces contextes avaient deux tailles : 10 observations et 5 attributs, puis 15 observations et 10 attributs. Pour chaque taille, trois densités ont été testées : 20%, 50% et 80%. Enfin, pour chaque configuration, 1000 contextes ont été générés.

Le treillis de concepts de chaque contexte a été calculé. Pour chaque couple de concepts (C_1, C_2) avec C_1 prédécesseur immédiat de C_2, la plus petite tolérance T telle que $C_1 T C_2$ a été calculée avec l'algorithme 4. Nous avons finalement testé si la tolérance était triviale, i.e. $T = O \times A$.

Les résultats de ces expériences sont donnés dans les figures 6.18 à 6.23.

Nous pouvons maintenant faire les observations suivantes :

- Le nombre de tolérances non-triviales augmente avec la densité.
- Mais, il semble décroître avec la taille du contexte.

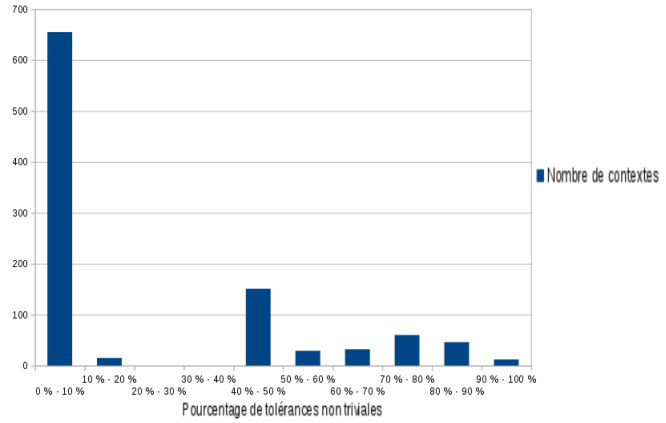


FIGURE 6.18 : Expérience avec les contextes de taille 10×5, densité 20%.

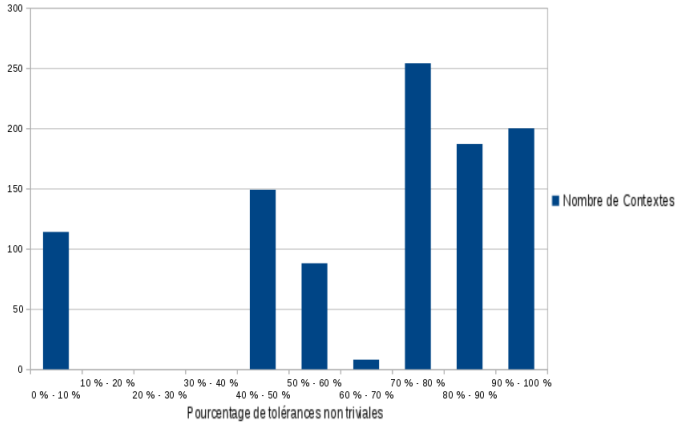


FIGURE 6.19 : Expérience avec les contextes de taille 10×5, densité 50%.

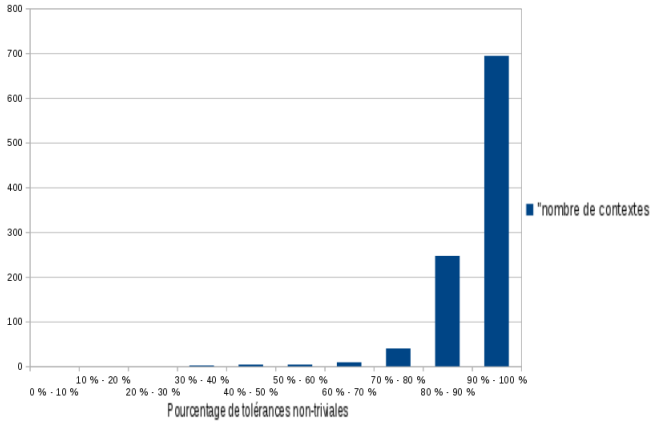


FIGURE 6.20 : Expérience avec les contextes de taille 10×5, densité 80%.

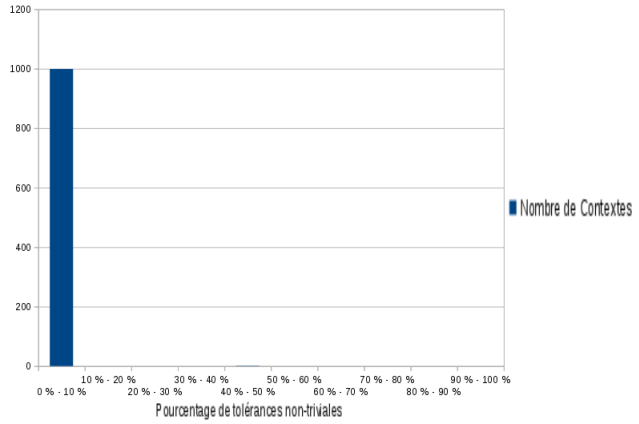


FIGURE 6.21 : Expérience avec les contextes de taille 15×10, densité 20%.

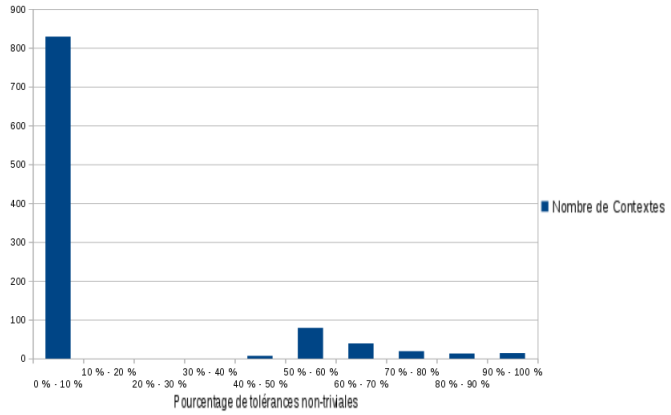


FIGURE 6.22 : Expérience avec les contextes de taille 15×10, densité 50%.

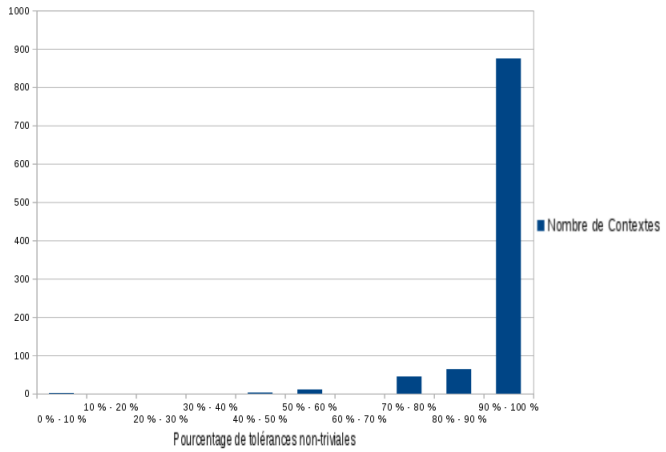


FIGURE 6.23 : Expérience avec les contextes de taille 15×10, densité 80%.

- Avec une densité faible, il n'y a approximativement pas de tolérance, mais avec une densité forte, il y en a beaucoup.
- Il semble y avoir un seuil, dépendant de la densité et de la taille, en-dessous duquel il n'y a pas de tolérance (non-triviale) et au-dessus duquel il y en a beaucoup.
- Globalement, il y a significativement plus de tolérances que de congruences. Cependant, il semble qu'il n'y ait pas assez de tolérances pour offrir une navigation souple pour notre projet LattExp. Nous devrons donc encore assouplir les conditions. La décomposition verticale peut toujours être envisagée en lien avec les *nested line diagrams*.

Troisième partie

Conclusions

Chapitre 7

Le mot de la fin

Sommaire

7.1 Contributions

Nous avons étudié les relations de congruence. En particulier, la décomposition sous-directe, à travers ses treillis quotients, a été décrite comme étape initiale pour une navigation. Ces relations de congruence ont également été utilisées pour définir une nouvelle décomposition, la construction de doublement inversée, basée sur la construction de doublement d'A. Day. Et finalement, les relations de congruence ont été utilisées pour obtenir des implications.

Ensuite, nous avons étudié les relations de tolérance, mis au point un algorithme permettant de calculer une tolérance particulière, et un autre pour calculer tout le treillis des tolérances. Ce treillis a ensuite été utilisé comme une

carte de navigation à la base du projet LattExp, lui-même a ensuite été utilisé pour analyser un réseau social (Section 6.1).

Nous avons également mis en place des algorithmes distribués utilisant Hadoop pour la FCA : le calcul d'un concept, des règles d'implication, de tout le treillis et d'une base d'implications.

Enfin, nous avons effectué des expériences sur les relations de congruences et les relations de tolérances sur des données tests afin d'avoir une idée générale sur la richesse de leur présence.

7.2 Cas d'usage de LattExp

L'Analyse de Concepts Formels permet l'étude des réseaux sociaux et l'étude de L. C. Freeman [73] fait apparaître que c'est une méthode qui fonctionne bien. Elle permet de :

- détecter des communautés,
- déterminer la place de leurs membres, dans le noyau, en périphérie, ou en lien avec d'autre communautés,
- visualiser grâce au treillis de concepts.

Cependant, nous avons vu que la lecture du treillis de concepts n'est pas toujours aisée. Lors de ses différentes réductions avec LattExp (Section 6.1) la lecture s'en est trouvée simplifiée et d'autres analyses ont été possibles, comme la découverte de sous-treillis isomorphes révélant une sorte d'équivalence structurelle, problématique importante de l'analyse de réseaux sociaux.

Le projet LattExp semble donc révéler des qualités pour l'analyse de réseaux sociaux. Nous allons donc nous en servir plus amplement, en particulier avec Twitter et FaceBook. Cela permettra de renforcer ses qualités et de corriger ses faiblesses. En particulier, le passage à l'échelle est un point critique.

7.3 Perspectives

Les perspectives pour améliorer LattExp sont nombreuses, à commencer par les visualisations qu'il produit. On peut identifier au moins deux directions :

- Enrichir les capacités de navigation.
- Définir un langage de requête.
- Améliorer le rendu visuel.

Concernant le premier point, là encore il y a au moins deux approches. La première est immédiate et consiste à améliorer les décompositions lesquelles sont principalement basées, au départ, sur un calcul sur le contexte lui-même. En effet la première étape consiste en un calcul sur les données. Ainsi, pour avoir de

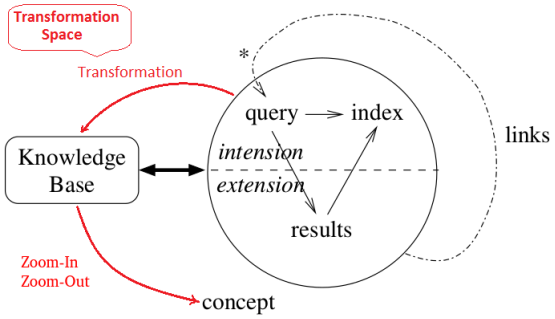


FIGURE 7.1 : LattExp vu comme une instance d'ACN enrichie d'une fonctionnalité de zoom.

meilleures décompositions/outils, et pour acquérir un meilleur savoir à partir des données, les opérations sur les contextes devraient être étudiées en profondeur, comme c'est le cas de l'algèbre relationnelle pour la manipulation des bases de données. Des études dans cette direction existent déjà dans la littérature [133, 155]. En particulier, nous allons étudier les algorithmes incrémentaux en lien avec l'opération de suppression, et l'opération consistant juste à ajouter/enlever une croix. Une étude systématique des liens entre, d'une part les implications et d'autre part les opérations sur les contextes [174, 171, 144], pourrait être faite.

La seconde approche pour améliorer les capacités de navigation, mais qui est liée à la première, consiste à utiliser le treillis des sous-treillis à la place du treillis des tolérances. Celui-ci n'est pas calculable ; cependant, il n'est pas utile d'avoir la carte du monde pour naviguer, donc ce n'est pas un problème de ne pas pouvoir calculer le treillis des sous-treillis. Et, d'un autre côté, ce treillis étant vraiment énorme, il offre une navigation très souple. Cette approche est liée à la première car les sous-treillis devront être identifiés à l'aide d'opérations sur le contexte initial. Avec l'étude empirique de G. Snelting [164] qui montre que beaucoup de contextes sont irréductibles pour la décompostion sous-directe, il est nécessaire de trouver des décompositions plus souples (par exemple les *NLD*) ou de combiner avec d'autres décompositions, en particulier en faisant intervenir la congruence de Fratini [59] ou la décomposition Atlas [88] qui utilise des tolérances. Il est également possible de sortir du cadre de la FCA [54].

En reprenant la figure 7.1 basée sur l'ACN et augmentée de la fonctionnalité de zoom/dézoom apportée par LattExp, on peut visualiser cette direction d'amélioration dans la partie "Espace des transformations". Pour le moment, cet espace des transformations correspond à l'ensemble des relations de congruences ou de tolérances. Il est trop petit et offre donc des fonctionnalités de zoom/dézoom limitées. En agrandissant cet espace, par exemple en considérant le treillis des sous-treillis [1], la fonctionnalité de transformation par zoom/dézoom apportée

[1] Il n'est pas envisageable de générer tout ce treillis, mais il est tout à fait possible de naviguer dedans.

par LattExp deviendrait bien plus souple et donc bien plus agréable pour l'utilisateur.

D'un point de vue technique, il est envisageable de mixer les deux approches présentées : calculer des décompositions de manière distribuée de sorte que chaque noeud du cluster calcule une partie de la décomposition et que le cluster entier ait accès à la totalité du treillis.

La seconde direction que nous avons mentionnée concerne les langages de requêtes. En effet, c'est une composante fondamentale dans la recherche d'informations et il n'y a pas de langage canonique de requête d'un treillis. Cependant, il est connu par au moins une partie de la communauté "floue" de la FCA, qu'un treillis résidué est capable d'encoder une logique du premier ordre et le langage associé [16]. Donc en déterminant, à partir d'un treillis donné, un treillis résidué, on obtiendrait un langage de requête. Or il s'avère que le treillis des tolérances d'un treillis est résidué [16].

7.4 Conclusion générale

Parmi les branches applicatives dont le développement actuel (2016) est très actif, il est important de citer le web sémantique [70]. De nombreux travaux récents lient la FCA et les ontologies [166], les logiques de descriptions [138, 150], le langage SPARQL [70, 6], etc. et ouvrent des directions de recherche prometteuses.

Ces travaux se situent à l'échelle du Web, donc des outils basés sur une décomposition quelconque peuvent s'avérer utiles.

Annexes

Sommaire

7.5 Exemples

Quelques-uns des exemples présentés ici sont disponibles sur le site Web d'U. Priss ainsi que leur origine [154].

7.5.1 La vie dans l'eau

Le contexte de cet exemple est donné par le tableau 7.1.

La figure 7.2 donne le treillis de concepts de ce contexte.

Cet exemple a juste pour objectif d'illustrer quelques points très simples que permet la FCA, en utilisant la figure 7.2 :

- Quelles sont les caractéristiques de la sangsue aquatique (*fish leech*) ? Il suffit de regarder les attributs [2] qui sont au-dessus, donc elle peut bouger, vit dans l'eau et a besoin d'eau pour vivre.

[2] En toute rigueur, il s'agit des concept-attributs situés au-dessus du concept-objet de la sangsue

	a	b	c	d	e	f	g	h	i
sangsue (*fish leech*)	x	x	↑	↑			x	↕	
dorade (*bream*)	x	x	↕	↑			x	x	
grenouille (*frog*)	x	x	x	↕	↓	↓	x	x	↕
chien (*dog*)	x	↕	x	↕	↓	↓	x	x	x
algue (*water weeds*)	x	x	↕	x		x	↑		
roseau (*reed*)	x	x	x	x	↕	x	↕	↓	↓
haricot (*bean*)	x	↕	x	x	x	↕	↕	↓	↓
maïs (*corn*)	x	↕	x	x	↑	x	↑		

TABLE 7.1 : La signification des attributs est la suivante : a-a besoin d'eau pour vivre, b-aquatique, c-terrestre, d-a besoin de chlorophille, e-deux cotylédons, f-un cotylédon, g-se déplace, h-a des membres, i-allaite

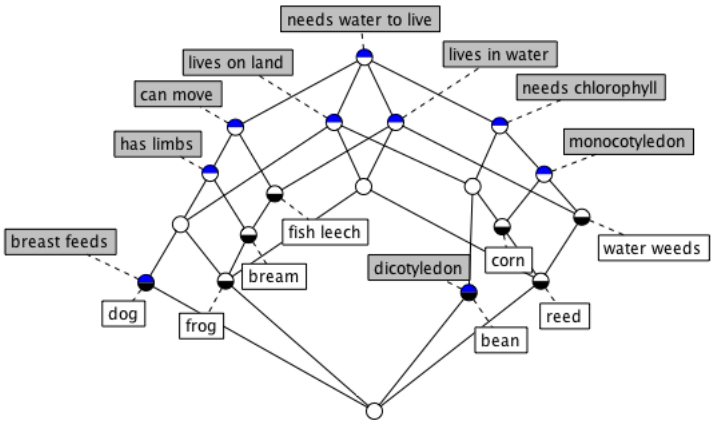


FIGURE 7.2 : Treillis de concepts du contexte 7.1

- Quels sont les points communs entre la sangsue et le chien ? On va chercher le noeud le plus bas se situant au-dessus, i.e le supremum, puis tout ce qui est au-desuss et ses attributs sont : se déplace et a besoin d'eau pour vivre.
- Y a-t-il d'autres objets qui possèdent toutes les caractéristiques du maïs ? En dessous, nous trouvons le roseau (*reed*) ; en plus des caractéristiques du maïs, il vit aussi dans l'eau.
- Peut-on envisager une créature ayant les caractéristiques du chien et nécessitant de la chlorophyle ? Dans le tableau, il y a une double flèche et le supremum de ces deux concepts est le supremum du treillis ; cela signifie que cette créature est encore moins probable qu'un ornithorynque.
- Quelles sont les caractéristiques minimales permettant d'identifier une sangsue ? Elle peut bouger et vit dans l'eau. Le besoin en eau pour vivre étant commun à tous, il n'est pas discriminant.
- Même question pour le chien ? C'est le seul qui allaite. Nous avons ici la problématique des générateurs minimaux.

Il est bien sûr possible d'allonger beaucoup cette liste d'illustrations. Terminons seulement en donnant des implications. Les bases d'implication (Guigues et Duquenne d'une part et Canonique Directe d'autre part) sont données dans les tableaux 7.3 et 7.2.

Enfin, d'un point de vue des relations de tolérance ou de congruence. Ce treillis ne possède qu'une seule relation non-triviale qui est une relation de congruence. Donc ici, toutes les relations de tolérance sont aussi des relations de congruence.

7.5.2 Quelques couleurs

Considérons la représentation en RVB des onzes couleurs suivantes :

- White $(1; 1; 1)$
- Grey $(0.5; 0.5; 0.5)$
- Black $(0; 0; 0)$
- Red $(1; 0; 0)$
- Orange $(1; 0.5; 0)$
- Yellow $(1; 1; 0)$
- Green $(0; 1; 0)$
- Blue $(0; 0; 1)$
- Purple $(0.5; 0; 0.5)$

"un cotylédon,besoin eau"→"besoin chlorophyle"
"terrestre,un cotylédon"→"besoin chlorophyle"
"aquatique,besoin chlorophyle"→"un cotylédon"
"aquatique,un cotylédon"→"besoin chlorophyle"
"a des membres,besoin eau"→"se déplace"
"a des membres,besoin chlorophyle"→"allaite,se déplace,deux cotyledons,aquatique,terrestre,un cotylédon"
"a des membres,un cotylédon"→"allaite,se déplace,deux cotyledons,aquatique,terrestre,besoin chlorophyle"
"a des membres,terrestre"→"se déplace"
"a des membres,aquatique"→"se déplace"
"deux cotyledons,besoin eau"→"terrestre,besoin chlorophyle"
"deux cotyledons,besoin chlorophyle"→"terrestre"
"deux cotyledons,un cotylédon"→"allaite,se déplace,a des membres,aquatique,terrestre"
"deux cotyledons,terrestre"→"besoin chlorophyle"
"deux cotyledons,aquatique"→"allaite,se déplace,a des membres,terrestre,un cotylédon,besoin chlorophyle"
"deux cotyledons,a des membres"→"allaite,se déplace,aquatique,terrestre,un cotylédon,besoin chlorophyle"
"se déplace,besoin chlorophyle"→"allaite,deux cotyledons,a des membres,aquatique,terrestre,un cotylédon"
"se déplace,un cotylédon"→"allaite,deux cotyledons,a des membres,aquatique,terrestre,besoin chlorophyle"
"se déplace,terrestre"→"a des membres"
"se déplace,deux cotyledons"→"allaite,a des membres,aquatique,terrestre,un cotylédon,besoin chlorophyle"
"allaite,besoin eau"→"se déplace,a des membres,terrestre"
"allaite,besoin chlorophyle"→"se déplace,deux cotyledons,a des membres,aquatique,terrestre,un cotylédon"
"allaite,un cotylédon"→"se déplace,deux cotyledons,a des membres,aquatique,terrestre,besoin chlorophyle"
"allaite,terrestre"→"se déplace,a des membres"
"allaite,aquatique"→"se déplace,deux cotyledons,a des membres,terrestre,un cotylédon,besoin chlorophyle"
"allaite,a des membres"→"terrestre"
"allaite,deux cotyledons"→"se déplace,a des membres,aquatique,un cotylédon,besoin chlorophyle"
"allaite,se déplace"→"a des membres,terrestre"

TABLE 7.2 : La base canonique directe du contexte 7.1

$\emptyset \rightarrow$ "besoin eau"
"aquatique,besoin chlorophyle" $\rightarrow$ "un cotylédon"
"deux cotylédons" $\rightarrow$ "terrestre,besoin chlorophyle"
"un cotylédon" $\rightarrow$ "besoin chlorophyle"
"terrestre,se déplace" $\rightarrow$ "a des membres"
"a des membres" $\rightarrow$ "se déplace"
"allaite" $\rightarrow$ "terrestre,se déplace,a des membres"
"aquatique,terrestre,se déplace,a des membres,allaite" $\rightarrow$ "besoin chlorophyle,deux cotylédons,un cotylédon"
"terrestre,besoin chlorophyle,deux cotylédons,un cotylédon" $\rightarrow$ "aquatique,se déplace,a des membres,allaite"
"besoin chlorophyle,se déplace" $\rightarrow$ "aquatique,terrestre,deux cotylédons,un cotylédon,a des membres,allaite"

TABLE 7.3 : La base de Guigues et Duquenne du contexte 7.1

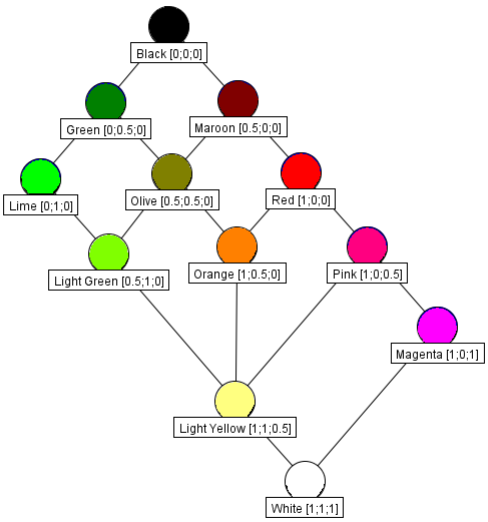


FIGURE 7.3 : Diagramme de Hasse de l'ensemble des couleurs

- Maroon $(0.5; 0; 0)$
- Pink $(1; 0; 0.5)$

On dira qu'une couleur est plus petite qu'une autre lorsque toutes ses composantes RVB sont plus petites que celles de l'autres.

Le diagramme de Hasse de cet ensemble ordonné est donné sur la figure 7.3.

Cet exemple a été beaucoup utilisé dans ce document.

7.5.3 Le système solaire

Considérons le contexte de la table 7.4

Remarques : Cet exemple est issu de [181]. Les attributs *Proche* et *Loin* sont relatifs à la distance au soleil.

Pour l'actualiser, il conviendrait de retirer Pluton, si nous étions des puristes spécialistes :-) ; pour cela, il est intéressant d'utiliser un algorithme incrémental capable de gérer un « delete ».

Ce contexte est un cas particulier d'une situation qui semble plus générale. Il est issu d'un contexte multivalué et donc ses attributs sont groupés : chaque observation possède un et un seul attribut dans un groupe. Il semble que les contextes ayant cette particularité ne possèdent pas de relations non triviales ni de congruence, ni de tolerance. C'est le cas de cet exemple.

	Petite	Moyenne	Grande	Proche	Loin	Lune	Pas Lune
Mercure	x			x			x
Venus	x			x			x
Terre	x			x		x	
Mars	x			x		x	
Jupiter			x		x	x	
Saturne			x		x	x	
Uranus		x			x	x	
Neptune		x			x	x	
Pluton	x				x	x	

TABLE 7.4 : Contexte binaire issu du contexte multivalué des planètes

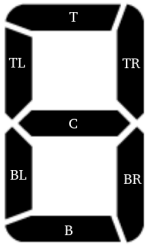


FIGURE 7.4 : Sémantique des attributs du context des chiffres digitaux.

7.5.4 Chiffres digitaux

Cet exemple utilise le contexte donné dans la table 7.5. La sémantique est la suivante. Tous les chiffres peuvent être affichés à l'aide de 7 diodes : en Haut (Top : T), au Centre (Center : C), en Bas (Bottom : B), en Haut à Gauche (Top Left : TL), en Haut à droite (Top Right : TR), en Bas à Gauche (Bottom Left : BL) ou en Bas à Droite (Bottom Right : BR). Par exemple, le chiffre 1 s'affiche juste avec "Top Right" and "Bottom Right", alors que le chiffre 0 a besoin de tout allumer sauf le centre (Figure 7.4).

La base canonique de ce contexte est : $TL \rightarrow BR$, $BL \rightarrow B$, $B - TR \rightarrow T$, $T - B - BL \rightarrow TR$, $B - BL - BR \rightarrow TL$, $T - B - TL - TR - BR \rightarrow BL$

La base canonique directe est : $TL \rightarrow BR$, $BL \rightarrow B$, $BL - TR \rightarrow T$, $BL - T \rightarrow TR$, $BL - BR \rightarrow TL$, $B - TR \rightarrow T$, $B - TL - TR \rightarrow BL$, $B - T - TL - TR \rightarrow BL$

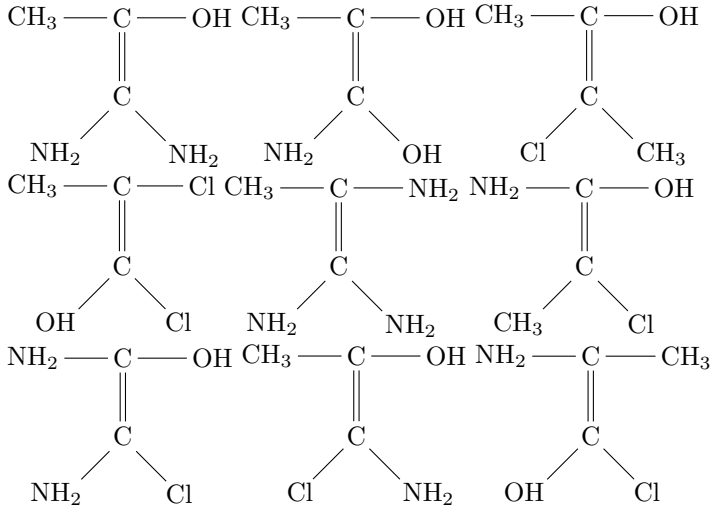
Le treillis de concepts possède 48 noeuds. Cet exemple est intéressant du point de vue des relations car le treillis des congruences possède 5 noeuds, et le trellis des tolérances en possède 10. Il y a donc possibilité de naviguer.

	T	C	B	TL	BL	TR	BR
0	x		x	x	x	x	x
1						x	x
2	x	x	x		x	x	
3	x	x	x			x	x
4		x		x		x	x
5	x	x	x	x			x
6		x	x	x	x		x
7	x					x	x
8	x	x	x	x	x	x	x
9	x	x		x		x	x

TABLE 7.5 : Contexte des chiffres digitaux. T : Top, C : Center, B : Bottom, TL : Top Left, BL : Bottom Left, TR : Top Right, BR : Bottom Right.

7.5.5 Quelques molécules

La généralisation de la FCA utilisant les pattern structures est capable de travailler avec ce genre de données complexes [120, 116] :



7.5.6 IPAQ

L'exemple suivant est basé sur les données de l'IPAQ (International Physical Activity Questionnaire) [18].

A nouveau ce contexte est déduit d'un contexte multivalué et il est irréductible, c'est à dire qu'il n'a pas de relations non triviales aussi bien de congruence que de tolérance.

	H	F	E.O	E.N	15-24	25-34	35-44	45-54	55-65	IMC<	IMC=	IMC>	Act<	Act≥
1	x		x					x				x	x	
2	x		x		x						x		x	
3	x		x					x			x		x	
4	x		x					x			x		x	
5	x		x					x			x		x	
6	x			x			x			x			x	
7	x			x	x						x		x	
8	x			x				x			x		x	
9	x			x				x			x		x	
10		x	x		x					x			x	
11		x	x				x				x		x	
12		x	x		x						x		x	
13		x	x		x					x			x	
14		x	x		x					x			x	
15		x	x			x				x			x	
16		x		x					x	x			x	
17		x		x	x					x			x	
18		x		x	x					x			x	
19		x		x	x					x			x	

TABLE 7.6 : Contexte issu des données de l'IPAQ

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Evelyn	x	x	x	x	x	x		x	x					
Laura	x	x	x		x	x	x	x						
Theresa		x	x	x	x	x	x	x	x					
Brenda	x		x	x	x	x	x	x						
Charlotte			x	x	x		x							
Frances			x		x	x		x						
Eleanor					x	x	x	x						
Pearl						x		x	x					
Ruth					x		x	x	x					
Verne							x	x	x			x		
Myra								x	x	x		x		
Katherine								x	x	x		x	x	x
Sylvia							x	x	x	x		x	x	x
Nora						x	x		x	x	x	x	x	x
Helen							x	x		x	x	x		
Dorothy								x	x					
Olivia									x		x			
Flora									x		x			

TABLE 7.7 : Participation des femmes aux évènements

7.5.7 Femmes de Caroline du Sud

Dans le jeu de données de la table 7.5.7, des femmes de Caroline du Sud participaient à des évènements.

D'après L. C. Freeman [74], ce jeu de données a été créé aux cours des années 30 par les ethnographes : A. Davis, E. Stubbs Davis, B. B. Gardner, M. R. Gardner and J. G. St Clair Drake.

L. C. Freeman a beaucoup étudié ce jeu de données [74, 73, 75], en particulier il a fait une étude comparative de différentes méthodes pour détecter les groupes sociaux. Il a ainsi fait une méta-classification ; il a classifié les méthodes de classification, en particulier, la FCA apparaît comme une bonne méthode sur ce type de données.

Cet exemple a également été utilisé par R. Wille [186], en particulier sa base canonique a été calculée.

Dans la section 6.1, un jeu de données a été extrait de celui-ci pour illustrer le fonctionnement de notre projet.

7.6 Quelques logiciels de FCA

Un tour d'horizon complet des logiciels de FCA existe déjà[168], celui-ci est plus récent mais moins complet.

- ConExp[192] : Probablement le plus utilisé (2016) par la communauté FCA.
- ConImp[30], FCALib et Conexp-clj : des librairies généralistes pour la FCA.
- Galicia et LatticeMiner : Développés par l'équipe canadienne.
- GLAD : Probablement le plus connus à cause de ses magnifiques visualisations. Développé par V. Duquenne. Non disponible.
- FCART : Récemment (16/03) rendu téléchargeable. Développé par l'équipe russe. Logiciel très prometteur.
- FCAStone : développé par U. Priss, capable de convertir les données.
- LatDraw : développé par R. Freese, fournissant un algorithme original de visualisation.
- TheGalactic : développé par l'équipe rochelaise. C'est une libraire généraliste pour la FCA. Le contenu de cette thèse y sera bientôt disponible. LattExp est une part de ce projet.
- ToscanaJ : un des plus anciens et utilisé par K. E. Wolff pour étudier l'évolution temporelle des données.
- LatViz : en ligne, développé au LORIA.

7.7 Questions

Q (Vincent Duquenne) : Est-il possible de pondérer les implications par leur support et d'alors développer un système de calcul propositionnel non-transitif ?

Q : Y a-t-il autre chose à fouiller dans les données que des concepts et des règles ? Les symétries internes ou les équivalences structurelles ?

Q (Vincent Duquenne) : Peut-on traduire les problèmes de relations de dépendances (D, δ, etc) en termes de relations flèches ?

Q : Y a-t-il un lien entre la décomposition subtensorielle et les jointures dans les bases de données.

Q (Amedeo Napoli) : Il faudrait formaliser le lien entre la notion de concept et la notion de sémantique en logique.

Q : Est-ce que la théorie des faisceaux peut être utilisée pour recoller des contextes, des treillis, des implications ?

Q (Karell Bertet) : Y a-t-il un algorithme polynomial permettant de calculer le contexte à partir de la base canonique (sans passer par le treillis) ?

R (Lhouari Nourine) : Je pense NP-complet.

Q : Peut-on calculer un(des) quotient(s) à l'aide d'un seul job Hadoop ?

Q (Karell Bertet) : Y a-t-il un algorithme polynomial de génération de la base canonique ou de la base canonique directe à partir d'un opérateur de fermeture ?

Q (Rokia Missaoui) : En considérant les décompositions comme des méthodes de réduction de dimension, peut-on leur trouver un lien avec l'analyse en composantes principales ou la décomposition en valeurs singulières ?

Q (Karell Bertet) : Il y a un lien connu entre la relation de dépendance et la base canonique directe ordonnée, et un autre lien entre la relation de dépendance faible et la base canonique directe. Y a-t-il une relation pour la base canonique ?

Q (Rokia Missaoui, Amedeo Napoli) : Est-il possible de définir un algorithme incrémental permettant le calcul d'une décomposition ? ce calcul se faisant en même temps qu'une construction incrémentale du treillis.

Q (Leonard Kwuida) : Peut-on définir un cadre général unifiant les méthodes statistiques et les méthodes algébriques ? De manière plus spécifique, une décomposition verticale conduisant à un *nested line diagram* basée sur des attributs choisis à partir d'une PCA est-elle pertinente ?

Q (Sergei O. Kuznetsov) : Les projections dans le cadre des *pattern structures* et les tolérances peuvent être considérées, pour les contextes, comme des mécanismes de simplification. Quels liens y a-t-il entre ces deux mécanismes ?

Q (Leonard Kwuida) : Peut-on définir un langage de requête (LQL : Lattice Query Langage) permettant à l'utilisateur de définir une première réduction initiant la navigation ?

Bibliographie

[1] Kira V. Adaricheva and James B. Nation. Discovery of the d-basis in binary tables based on hypergraph dualization. *CoRR*, abs/1504.02875, 2015. p 69

[2] Kira V. Adaricheva, James B. Nation, Gordon Okimoto, Vyacheslav Adariche va, Adina Amanbekkyzy, Shuchismita Sarkar, Alibek Sailanbayev, Nazar Seidalin, and Kenneth Alibek. Measuring the implications of the d-basis in analysis of data in biomedical studies. In Jaume Baixeries, Christian Sacarea, and Manuel Ojeda-Aciego, editors, *Formal Concept Analysis*, volume 9113 of *Lecture Notes in Computer Science*, pages 39–57. Springer International Publishing, 2015. p 39, p 40, p 45, p 87

[3] Kira V. Adaricheva, James B. Nation, and Robert Rand. Ordered direct implicational basis of a finite closure system. In *ISAIM*, 2012. p 39, p 40, p 68, p 87

[4] P. Agarwal, Mehdi Kaytoue, Sergei O. Kuznetsov, Amedeo Napoli, and Géraldine Polaillon. Symbolic galois lattices with pattern structures. In Sergei O. Kuznetsov, Dominik Slezak, Daryl H. Hepting, and Boris G. Mirkin, editors, *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, volume 6743 of *Lecture Notes in Computer Science*, pages 191–198. Springer Berlin Heidelberg, 2011. p 44

[5] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. p 14

[6] Mehwish Alam. *Interactive Knowledge Discovery over Web of Data.* Theses, Loria & Inria Grand Est, December 2015. p 3, p 13, p 120

[7] M.R. Amini and E. Gaussier. *Recherche d'information : Applications, modèles et algorithmes - Fouille de données, décisionnel et big data.* Algorithmes. Eyrolles, 2013. p 4

[8] Marta Arias and José-Luis. Balcázar. Construction and learnability of canonical horn formulas. *Machine Learning*, 85(3) :273–297, 2011. p 41

[9] William Ward Armstrong. Dependency structures of data base relationships. In *IFIP Congress*, pages 580–583, 1974. p 38

[10] Antoine Arnauld and Pierre Nicole. *La logique ou L'art de penser / Antoine Arnauld, Pierre Nicole ; notes et postf. de Charles Jourdain.* Gallimard (Paris), 1992. p 5, p 17

[11] Jaume Baixeries and José Luis Balcázar. New closure operators and lattice representations for multivalued dependencies and related expressions. In *The 3rd international conference on Concept Lattices and Their Applications (CLA'05)*, 2005. p 37

[12] Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli. Computing functional dependencies with pattern structures. In *Proceedings of The Ninth International Conference on Concept Lattices and Their Applications, Fuengirola (Málaga), Spain, October 11-14, 2012*, pages 175–186, 2012. p 44

[13] Fatma Baklouti and Gérard Lévy. Distributed and general galois lattice for large data bases. In *The 3rd international conference on Concept Lattices and Their Applications (CLA'05)*, 2005. p 41

[14] Hans-J Bandelt. Tolerance relations on lattices. *Bulletin of the Australian Mathematical Society*, 23(03) :367–381, 1981. p 33

[15] Marc Barbut and Bernard Monjardet, editors. *L'ordre et la classification.* Algèbre et combinatoire, tome I et II. Hachette, 1970. p 5, p 18, p 19, p 26, p 28, p 45

[16] Eduard Bartl and Michal Krupka. Logical analysis of concept lattices by factorization. In *Formal Concept Analysis*, pages 16–27. Springer, 2012. p 44, p 120

[17] Radim Belohlavek. Concept lattices and order in fuzzy logic. *Annals of Pure and Applied Logic*, 128(1–3) :277 – 298, 2004. p 43

[18] Radim Belohlavek, Vladimır Sklenár, Jirı Zacpal, and Erik Sigmund. Evaluation of questionnaires by means of formal concept analysis. In *CLA*, pages 100–111, 2007. p 42, p 45, p 128

[19] Radim Belohlavek and Vilem Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *Journal of Computer and System Sciences*, 76(1) :3–20, February 2010. p 7

[20] Karell Bertet. *Sur quelques aspects algorithmiques et structurels des treillis.* PhD thesis, Université de Paris VII, 1998. p 29

[21] Karell Bertet. *Structure de treillis : contributions structurelles et algorithmiques : quelques usages pour des données images.* Hdr, Université de La Rochelle, 2010. p 29

[22] Karell Bertet. The dependence graph of a lattice. In *Proceedings of The Ninth International Conference on Concept Lattices and Their Applications, Fuengirola (Málaga), Spain, October 11-14, 2012*, pages 223–232, 2012. p 40

[23] Karell Bertet, Daniel Borchmann, Peggy Cellier, and Sébastien Ferré, editors. *Formal Concept Analysis - 14th International Conference, ICFCA 2017, Rennes, France, June 13-16, 2017, Proceedings*, volume 10308 of *Lecture Notes in Computer Science*. Springer, 2017. p 45

[24] Karell Bertet and Nathalie Caspard. Doubling convex sets in lattices : characterizations and recognition algorithms. Technical Report TR-LACL-2002-08, LACL (Laboratory of Algorithms, Complexity and Logic), University of Paris-Est (Paris 12), 2002. p 7, p 70

[25] Karell Bertet and Bernard Monjardet. The multiple facets of the canonical direct unit implicational basis. *Theoretical Computer Science*, 411(22–24) :2155 – 2166, 2010. p 39, p 40

[26] Garett Birkhoff. Lattice theory. In *Colloquium Publications*, volume 25. American Mathematical Society, 1967. Third edition. p 5

[27] Jean-Pierre Bordat. Calcul pratique du treillis de galois d'une correspondance. *Mathématiques et Sciences Humaines*, 96 :31–47, 1986. p 41

[28] C. Brasseur. *Enjeux et usages du Big Data : Technologies, méthodes et mise en oeuvre*. Collection Management et informatique. Hermes Science Publications, 2013. p 4

[29] Paula Brito and Géraldine Polaillon. Structuring probabilistic data by galois lattices. *Mathématiques et sciences humaines. Mathematics and social sciences*, 169, 2005. p 44

[30] Peter Burmeister. *Formal concept analysis with ConImp : Introduction to the basic features*. Darmstadt University of Technology, 2003. p 131

[31] Claudio Carpineto and Giovanni Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2) :95–122, 1996. p 42

[32] Claudio Carpineto and Giovanni Romano. *Concept Data Analysis : Theory and Applications*. Wiley InterScience online books. John Wiley & Sons, 2004. p 5, p 45, p 46

[33] Nathalie Caspard and Bernard Monjardet. The lattices of moore families and closure operators on a finite set : A survey. *Electronic Notes in Discrete Mathematics*, 2(0) :25 – 50, 1999. OSDA98, Ordinal and Symbolic Data Analysis. p 28

[34] Nathalie Caspard and Bernard Monjardet. The lattices of closure systems, closure operators, and implicational systems on a finite set : a survey. *Discrete Applied Mathematics*, 127(2) :241 – 269, 2003. Ordinal and Symbolic Data Analysis (OSDA '98), Univ. of Massachusetts, Amherst, Sept. 28-30, 1998. p 28, p 40

[35] Ivan Chajda and Bohdan Zelinka. Lattices of tolerances. *Časopis pro pěstování matematiky*, 102(1) :10–24, 1977. p 33, p 34

[36] Ivan Chajda and Bohdan Zelinka. Minimal compatible tolerances on lattices. *Czechoslovak Mathematical Journal*, 27(3) :452–459, 1977. p 33

[37] Richard Cole and Peter W. Eklund. Scalability in formal concept analysis. *Computational Intelligence*, 15(1) :11–27, 1999. p 41

[38] Pablo Cordero, Manuel Enciso, and Angel Mora. Automated reasoning to infer all minimal keys. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 817–823. AAAI Press, 2013. p 38

[39] François Coste, Gaëlle Garet, Agnès Groisillier, Jacques Nicolas, and Thierry Tonon. Automated enzyme classification by formal concept analysis. In *Formal Concept Analysis - 12th International Conference, ICFCA 2014, Cluj-Napoca, Romania, June 10-13, 2014. Proceedings*, pages 235–250, 2014. p 45

[40] Adrien Coulet, Florent Domenach, Mehdi Kaytoue, and Amedeo Napoli. Using pattern structures for analyzing ontology-based annotations of biomedical data. In Peggy Cellier, Felix Distel, and Bernhard Ganter, editors, *Formal Concept Analysis*, volume 7880 of *Lecture Notes in Computer Science*, pages 76–91. Springer Berlin Heidelberg, 2013. p 45

[41] Peter Crawley and Robert P. Dilworth. *Algebraic theory of lattices*. Prentice-Hall, 1973. p 5, p 29, p 32

[42] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines : Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA, 1st edition, 2009. p 12

[43] Gábor Czédli. Factor lattices by tolerances. *Acta Sci. Math.(Szeged)*, 44(1-2) :35–42, 1982. p 33

[44] Gábor Czédli and George Grätzer. Lattice tolerances and congruences. *Algebra universalis*, 66(1) :5–6, 2011. p 33

[45] Frithjof Dau and Julia Klinger. From formal concept analysis to contextual logic. In *Formal Concept Analysis*, pages 81–100. Springer, 2005. p 5

[46] Alan Day. A simple solution to the word problem for lattices. *Canad. Math. Bull*, 13 :253–254, 1970. p 7

[47] Alan Day. Splitting lattices generate all lattices. *Algebra universalis*, 7(1) :163–169, 1977. p 7, p 35

[48] Alan Day. Characterizations of finite lattices that are bounded-homomorphic images or sublattices of free lattices. *Canad. J. Math.*, 31 :69–78, 1979. p 7, p 32

[49] Alan Day. Congruence normality : The characterization of the doubling class of convex sets. *Algebra universalis*, 31(3) :397–406, 1994. p 7, p 35, p 37, p 69

[50] Alan Day, James B. Nation, and Steve Tschantz. Doubling convex sets in lattices and a generalized semidistributivity condition. *Order*, 6(2) :175–180, 1989. p 7, p 35

[51] Jiří Demel. Fast algorithms for finding a subdirect decomposition and interesting congruences of finite algebras. *Kybernetika (Prague)*, 18(2) :121–130, 1982. p 7

[52] Edwin Diday and Richard Emilion. Treillis de galois maximaux et capacités de choquet. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 325(3) :261 – 266, 1997. p 44

[53] Edwin Diday and Richard Emilion. Maximal and stochastic galois lattices. *Discrete Applied Mathematics*, 127(2) :271 – 284, 2003. Ordinal and Symbolic Data Analysis (OSDA '98), Univ. of Massachusetts, Amherst, Sept. 28-30, 1998. p 44

[54] Didier Dubois and Henri Prade. Formal concept analysis from the standpoint of possibility theory. In *Formal Concept Analysis - 13th International Conference, ICFCA 2015, Nerja, Spain, June 23-26, 2015, Proceedings*, pages 21–38, 2015. p 119

[55] Baptiste Ducatel, Mehdi Kaytoue, Florent Marcuola, Amedeo Napoli, and Laszlo Szathmary. Coron : Plate-forme d'extraction de connaissances dans les bases de données. In *17ème conférence en Reconnaissance des Formes et Intelligence Artificielle - RFIA 2010*, Reconnaissance des Formes et Intelligence Artificielle - RFIA 2010, pages 883–884, Caen, France, January 2010. p 45

[56] Vincent Duquenne. What can lattices do for experimental designs ? *Mathematical Social Sciences*, 11(3) :243–281, 1986. p 45

[57] Vincent Duquenne. Lattice analysis and the representation of handicap associations. *Social Networks*, 18(3) :217–230, 1996. p 45

[58] Vincent Duquenne. Latticial structures in data analysis. *Theor. Comput. Sci.*, 217(2) :407–436, 1999. p 5

[59] Vincent Duquenne. Lattice drawings and morphisms. In *Formal Concept Analysis, 8th International Conference, ICFCA 2010, Agadir, Morocco, March 15-18, 2010. Proceedings*, pages 88–103, 2010. p 42, p 119

[60] Vincent Duquenne. Congruence de treillis et classifications. *Société Française de Classification*, pages 21–24, 2011. p 42

[61] Vincent Duquenne. Contextual implications between attributes and some representation properties for finite lattices. In Peggy Cellier, Felix Distel, and Bernhard Ganter, editors, *ICFCA*, volume 7880 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 2013. p 17

[62] Vincent Duquenne and John Mohr. Comparison of dual orderings in time ii. In Raoul Medina and Sergei A. Obiedkov, editors, *ICFCA*, volume 4933 of *Lecture Notes in Computer Science*, pages 321–324. Springer, 2008. p 45

[63] Vincent Duquenne, John Mohr, and Annick Le Pape. Comparison of dual orderings in time. *Social science information*, 37(2) :227–253, 1998. p 45

[64] Vincent Duquenne and Bernard Monjardet. Relations binaires entre partitions. *Mathématiques et Sciences Humaines*, 80 :5–37, 1982. p 24

[65] Peter Eklund, Jon Ducrou, and Peter Brawn. *Concept lattices for information visualization : Can novices read line-diagrams ?* Springer, 2004. p 45, p 46

[66] Peter Eklund and Jean Villerd. A Survey of Hybrid Representation of Concept Lattices in Conceptual Knowledge Processing. In Léonard Kwuida and Baris Sertkaya, editors, *I8th International Conference on Formal Concept Analysis - ICFCA 2010*, volume 5986, pages 296–311, Agadir, Morocco, March 2010. Springer Berlin / Heidelberg. p 43

[67] Martin G Everett and Stephen P Borgatti. The dual-projection approach for two-mode networks. *Social Networks*, 35(2) :204–210, 2013. p 92

[68] Sébastien Ferré. *Systèmes d'information logiques : un paradigme logico-contextuel pour interroger, naviguer et apprendre.* PhD thesis, Rennes 1, 2002. p 44

[69] Sébastien Ferré. Negation, opposition, and possibility in logical concept analysis. In *Formal Concept Analysis*, volume 3874 of *Lecture Notes in Computer Science*, pages 130–145. Springer Berlin Heidelberg, 2006. p 44

[70] Sébastien Ferré. *Reconciling Expressivity and Usability in Information Access.* Habilitation à diriger des recherches, Université de Rennes 1, November 2014. p 8, p 9, p 12, p 13, p 14, p 15, p 42, p 44, p 46, p 47, p 120, p 151, p 153

[71] Sébastien Ferré and Olivier Ridoux. Une généralisation logique de l'analyse de concepts formels. Technical report, Inria, Institut National de Recherche en Informatique et en Automatique, December 1999. An english version is available at http ://www.irisa.fr/lande/ferre. p 44

[72] Sébastien Ferré and Olivier Ridoux. Introduction to logical information systems. *Information Processing & Management*, 40(3) :383–419, 2004. p 44

[73] Linton C. Freeman. Cliques, galois lattices, and the structure of human social groups. *Social Networks*, 18(3) :173 – 187, 1996. Social Network and Discrete Structure Analysis. p 45, p 118, p 130

[74] Linton C. Freeman. Finding social groups : A meta-analysis of the southern women data. In *Breiger, R., Carley, C., and Pattison, P.Dynamic Social Network Modeling and Analysis : Workshop Summary and Papers (pp 39-97), Washington D.C. : National Research Council, The National Academies*. Press, 2002. p 45, p 91, p 130

[75] Linton C. Freeman and Douglas R. White. Using galois lattices to represent network data. *Sociological Methodology*, 23 :127–146, 1993. p 45, p 91, p 130

[76] R Freese, J Jezek, and JB Nation. Free lattices (math. surv. monogr., 42), providence, ri. In *Am. Math. Soc*, volume 13, 1995. p 69

[77] Ralph Freese. Computing congruence lattices of finite lattices. *Proceedings of the American Mathematical Society*, 125(12) :3457–3463, 1997. p 7, p 32

[78] Ralph Freese. Algorithms for finite, finitely presented and free lattices. *Preprint, July*, 22 :159–178, 1999. p 7, p 32

[79] Ralph Freese. Automated lattice drawing. In *Concept Lattices, Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004, Proceedings*, pages 112–127, 2004. p 42

[80] Ralph Freese. Computing congruences efficiently. *Algebra universalis*, 59(3) :337–343, 2008. p 7, p 32

[81] Huaiguo Fu and Engelbert Mephu Nguifo. Partitioning large data to scale up lattice-based algorithm. In *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, pages 537–541, Nov 2003. p 41

[82] Huaiguo Fu and Engelbert Mephu Nguifo. A parallel algorithm to generate formal concepts for large data. In Peter Eklund, editor, *Concept Lattices*, volume 2961 of *Lecture Notes in Computer Science*, pages 394–401. Springer Berlin Heidelberg, 2004. p 41

[83] Petra Funk, Anke Lewien, and Gregor Snelting. Algorithms for concept lattice decomposition and their applications. Technical report, TU Braunschweig, December 1995. p 7, p 57

[84] Petr Gajdoš, Pavel Moravec, and Václav Snášel. Concept lattice generation by singular value decomposition. In *CLA*, volume 2004, pages 13–22, 2004. p 7

[85] Boris A. Galitsky, Sergei O. Kuznetsov, and Daniel Usikov. Parse thicket representation for multi-sentence search. In Heather D. Pfeiffer, Dmitry I. Ignatov, Jonas Poelmans, and Nagarjuna Gadiraju, editors, *Conceptual Structures for STEM Research and Education*, volume 7735 of *Lecture Notes in Computer Science*, pages 153–172. Springer Berlin Heidelberg, 2013. p 45

[86] Bernhard Ganter. Two basic algorithms in concept analysis. In Leonard Kwuida and Baris Sertkaya, editors, *Formal Concept Analysis*, volume 5986 of *Lecture Notes in Computer Science*, pages 312–340. Springer Berlin Heidelberg, 2010. p 39, p 41, p 64

[87] Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. In Harry S. Delugach and Gerd Stumme, editors, *Conceptual Structures : Broadening the Base*, volume 2120 of *Lecture Notes in Computer Science*, pages 129–142. Springer Berlin Heidelberg, 2001. p 43, p 44

[88] Bernhard Ganter and Rudolf Wille. *Formal concept analysis - mathematical foundations.* Springer, 1999. p 7, p 8, p 15, p 24, p 28, p 30, p 32, p 33, p 34, p 56, p 57, p 67, p 69, p 72, p 103, p 119

[89] Winfried Geyer. The generalized doubling construction and formal concept analysis. *Algebra universalis*, 32(3) :341–367, 1994. p 7, p 35

[90] Robert Godin, Rokia Missaoui, and Hassan Alaoui. Incremental concept formation algorithms based on galois (concept) lattices. *Computational Intelligence*, 11 :246–267, 1995. p 41

[91] George Grätzer. *General lattice theory.* Birkhäuser-Verlag, Basel, 1978. p 29

[92] George Grätzer. *The congruences of a finite lattice : a proof-by-picture approach.* Springer Science & Business Media, 2007. p 32

[93] George Grätzer. *Lattice Theory : Foundation.* SpringerLink : Bücher. Birkhauser Boston, 2011. p 5, p 41, p 69

[94] Joanna Grygiel. Distributive lattices with a given skeleton. *Discussiones Mathematicae, General Algebra and Applications*, 24 :75–94, 2004. p 33

[95] Joanna Grygiel and Sándor Radelecki. On the tolerance lattice of tolerance factors. *Acta Mathematica Hungarica*, 141(3) :220–237, 2013. p 74

[96] Clément Guérin, Karell Bertet, and Arnaud Revel. An efficient java implementation of the immediate successors calculation. In *Proceedings of the Tenth International Conference on Concept Lattices and Their Applications, La Rochelle, France, October 15-18, 2013.*, pages 81–92, 2013. p 41

[97] 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. p 39, p 64

[98] Joachim Hereth, Gerd Stumme, Rudolf Wille, and Uta Wille. *Conceptual knowledge discovery and data analysis*. Springer, 2000. p 17

[99] Lihua Hu, Jifu Zhang, and Sulan Zhang. A pruning based incremental construction of horizontal partitioned concept lattice. In De-Shuang Huang, Kang Li, and George William Irwin, editors, *Computational Intelligence*, volume 4114 of *Lecture Notes in Computer Science*, pages 936–945. Springer Berlin Heidelberg, 2006. p 41

[100] Yi-Chung Hu, Ruey-Shun Chen, and Gwo-Hshiung Tzeng. Discovering fuzzy association rules using fuzzy partition methods. *Knowledge-Based Systems*, 16(3) :137–147, 2003. p 39

[101] Marianne Huchard and Sergei Kuznetsov, editors. *Proceedings of the Thirteenth International Conference on Concept Lattices and Their Applications, Moscow, Russia, July 18-22, 2016*, volume 1624 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. p 45

[102] Marianne Huchard, Amedeo Napoli, Mohamed Rouane Hacene, and Petko Valtchev. Mining description logics concepts with relational concept analysis. In Paula Brito, Guy Cucumel, Patrice Bertrand, and Francisco de Carvalho, editors, *Selected Contributions in Data Analysis and Classification*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 259–270. Springer Berlin Heidelberg, 2007. p 43

[103] Mehdi Kaytoue, Sébastien Duplessis, and Amedeo Napoli. L'analyse formelle de concepts pour l'extraction de connaissances dans les données d'expression de gènes. In *Extraction et gestion des connaissances*, volume RNTI-E-15, pages 439–440, Strasbourg, France, January 2009. p 45

[104] Mehdi Kaytoue, Sergei O. Kuznetsov, Zainab Assaghir, and Amedeo Napoli. Embedding Tolerance Relations in Concept Lattices - An application in Information Fusion. Research Report RR-7353, Inria, Institut National de Recherche en Informatique et en Automatique, August 2010. p 74

[105] Mehdi Kaytoue, Sergei O. Kuznetsov, Juraj Macko, and Amedeo Napoli. Biclustering meets triadic concept analysis. *Annals of Mathematics and Artificial Intelligence*, 70(1-2) :55–79, 2014. p 43

[106] Mehdi Kaytoue, Sergei O. Kuznetsov, and Amedeo Napoli. Pattern Mining in Numerical Data : Extracting Closed Patterns and their Generators. Research Report RR-7416, Inria, Institut National de Recherche en Informatique et en Automatique, October 2010. p 44

[107] Mehdi Kaytoue, Sergei O. Kuznetsov, and Amedeo Napoli. Revisiting numerical pattern mining with formal concept analysis. *CoRR*, abs / 1111 . 5689, 2011. p 44

[108] Mehdi Kaytoue, Sergei O. Kuznetsov, Amedeo Napoli, and Géraldine Polaillon. Symbolic Data Analysis and Formal Concept Analysis. In Richard Emilion and Guillaume Cleuziou, editors, *XVIIIeme Rencontres de la Société Francophone de Classification - SFC 2011*, Orléans, France, September 2011. MAPMO - LIFO Orléans. p 44

[109] Petr Krajca, Jan Outrata, and Vilem Vychodil. Parallel recursive algorithm for fca. In *CLA*, volume 2008, pages 71–82, 2008. p 41

[110] Petr Krajca, Jan Outrata, and Vilem Vychodil. Advances in algorithms based on cbo. In *CLA*, pages 325–337, 2010. p 41

[111] Petr Krajca, Jan Outrata, and Vilem Vychodil. Parallel algorithm for computing fixpoints of galois connections. *Annals of Mathematics and Artificial Intelligence*, 59(2) :257–272, 2010. p 41

[112] Petr Krajca and Vilem Vychodil. Distributed algorithm for computing formal concepts using map-reduce framework. In Niall M. Adams, Céline Robardet, Arno Siebes, and Jean-François Boulicaut, editors, *Advances in Intelligent Data Analysis VIII*, volume 5772 of *Lecture Notes in Computer Science*, pages 333–344. Springer Berlin Heidelberg, 2009. p 78

[113] Francesco Kriegel and Daniel Borchmann. Nextclosures : Parallel computation of the canonical base. In *Proceedings of the Twelfth International Conference on Concept Lattices and Their Applications, Clermont-Ferrand, France, October 13-16, 2015.*, pages 181–192, 2015. p 41

[114] Sergei O. Kuznetsov. Pattern structures for analyzing complex data. In Hiroshi Sakai, Mihir Kumar Chakraborty, Aboul Ella Hassanien, Dominik Ślęzak, and William Zhu, editors, *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, volume 5908 of *Lecture Notes in Computer Science*, pages 33–44. Springer Berlin Heidelberg, 2009. p 44

[115] Sergei O. Kuznetsov. Fitting pattern structures to knowledge discovery in big data. In Peggy Cellier, Felix Distel, and Bernhard Ganter, editors,

Formal Concept Analysis, volume 7880 of *Lecture Notes in Computer Science*, pages 254–266. Springer Berlin Heidelberg, 2013. p 90

[116] Sergei O. Kuznetsov. Scalable knowledge discovery in complex data with pattern structures. In Pradipta Maji, Ashish Ghosh, M.Narasimha Murty, Kuntal Ghosh, and Sankark. Pal, editors, *Pattern Recognition and Machine Intelligence*, volume 8251 of *Lecture Notes in Computer Science*, pages 30–39. Springer Berlin Heidelberg, 2013. p 128

[117] Sergei O. Kuznetsov, Rokia Missaoui, and Sergei A. Obiedkov, editors. *Proceedings of the Workshop on Social Network Analysis using Formal Concept Analysis in conjunction with the 13th International Conference on Formal Concept Analysis (ICFCA 2015), Nerja (Malaga), Spain, June 25, 2015*, volume 1534 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015. p 45

[118] Sergei O. Kuznetsov and Sergei A. Obiedkov. Algorithms for the construction of concept lattices and their diagram graphs. In Luc De Raedt and Arno Siebes, editors, *Principles of Data Mining and Knowledge Discovery*, volume 2168 of *Lecture Notes in Computer Science*, pages 289–300. Springer Berlin Heidelberg, 2001. p 41

[119] Sergei O. Kuznetsov and Sergei A. Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of experimental and theoretical artificial intelligence*, 14 :189–216, 2002. p 41

[120] Sergei O. Kuznetsov and Mikhail V. Samokhin. Learning closed sets of labeled graphs for chemical applications. In Stefan Kramer and Bernhard Pfahringer, editors, *Inductive Logic Programming*, volume 3625 of *Lecture Notes in Computer Science*, pages 190–208. Springer Berlin Heidelberg, 2005. p 45, p 128

[121] Léonard Kwuida, Rokia Missaoui, Abdélilah Balamane, and Jean Vaillancourt. Generalized pattern extraction from concept lattices. *Annals of Mathematics and Artificial Intelligence*, 72(1-2) :151–168, 2014. p 44

[122] Gérard Lévy and Fatma Baklouti. A distributed version of the ganter algorithm for general galois lattices. In *CLA*, volume 2005, pages 207–221, 2005. p 41

[123] Leonid Libkin. Direct product decompositions of lattices, closures and relation schemes. *Discrete Mathematics*, 112(1) :119–138, 1993. p 7

[124] Michel Liquiere and Jean Sallantin. Structural machine learning with galois lattice and graphs. In Jude W. Shavlik, editor, *ICML*, pages 305–313. Morgan Kaufmann, 1998. p 44

[125] Estrella Rodríguez Lorenzo, Kira V. Adaricheva, Pablo Cordero, Manuel Enciso, and Angel Mora. From an implicational system to its corresponding d-basis. In *Proceedings of the Twelfth International Conference*

on Concept Lattices and Their Applications, Clermont-Ferrand, France, October 13-16, 2015., pages 217–228, 2015. p 40, p 68

[126] Ben Martin and Peter Eklund. Spatial indexing for scalability in fca. In Rokia Missaoui and Jürg Schmid, editors, *Formal Concept Analysis*, volume 3874 of *Lecture Notes in Computer Science*, pages 205–220. Springer Berlin Heidelberg, 2006. p 9

[127] Ralph McKenzie. Equational bases and nonmodular lattice varieties. *Transactions of the American Mathematical Society*, 174 :1–43, 1972. p 7

[128] Jesús Medina and Jorge Ruiz-Calviño. Fuzzy formal concept analysis via multilattices : First prospects and results. In *CLA*, pages 69–80, 2012. p 43

[129] Nizar Messai, Marie-Dominique Devignes, Amedeo Napoli, and Malika Smaïl-Tabbone. Connaissances de domaine et treillis de concepts pour l'exploration progressive de données complexes. In Sylvie Despres, editor, *21es Journées francophones d'Ingénierie des Connaissances - IC 2010*, pages 233 – 244, Nîmes, France, June 2010. Ecole des Mines d'Alès. p 74

[130] Ryszard S. Michalski and Robert E. Stepp. Automated construction of classifications : conceptual clustering versus numerical taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 5(4) :396–410, 1983. p 44

[131] Ryszard S. Michalski and Robert E. Stepp. Learning from observation : Conceptual clustering. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning*, Symbolic Computation, pages 331–363. Springer Berlin Heidelberg, 1983. p 44

[132] Peter Mihók and Gabriel Semanišin. Unique factorization theorem and formal concept analysis. In Sadok Ben Yahia, Engelbert Mephu Nguifo, and Radim Belohlavek, editors, *Concept Lattices and Their Applications*, volume 4923 of *Lecture Notes in Computer Science*, pages 232–239. Springer Berlin Heidelberg, 2008. p 7

[133] Rokia Missaoui, Léonard Kwuida, Mohamed Quafafou, and Jean Vaillancourt. Algebraic operators for querying pattern bases. *CoRR*, abs/0902.4042, 2009. p 119

[134] Rokia Missaoui, Lhouari Nourine, and Yoan Renaud. Computing implications with negation from a formal context. *Fundam. Inform.*, 115(4) :357–375, 2012. p 39

[135] Bernard Monjardet and Nathalie Caspard. On a dependence relation in finite lattices. *Discrete Mathematics*, 165–166(0) :497 – 505, 1997. Graphs and Combinatorics. p 40, p 41

[136] Angel Mora, Pablo Cordero, Manuel Enciso, Inmaculada Fortes, and Gabriel Aguilera. Closure via functional dependence simplification. *International Journal of Computer Mathematics*, 89(4) :510–526, 2012. p 38

[137] Angel Mora, Inmaculada P. de Guzmán, Manuel Enciso, and Pablo Cordero. Ideal non-deterministic operators as a formal framework to reduce the key finding problem. *International Journal of Computer Mathematics*, 88 :1860–1868, 2011. p 38

[138] Amedeo Napoli. Une introduction aux logiques de descriptions. Research Report RR-3314, Inria, Institut National de Recherche en Informatique et en Automatique, 1997. p 120

[139] James B. Nation. Alan day's doubling construction. *Algebra universalis*, 34(1) :24–34, 1995. p 7, p 35

[140] James B. Nation. Closure operators and lattice extensions. *Order*, 21(1) :43–48, 2004. p 64

[141] Josef Niederle. On atoms in tolerance lattices of distributive lattices. *Časopis pro pěstování matematiky*, 106(3) :311–315, 1981. p 33

[142] Josef Niederle. A note on tolerance lattices. *Časopis pro pěstování matematiky*, 107(3) :221–224, 1982. p 33, p 34

[143] Lhouari Nourine and Olivier Raynaud. A fast algorithm for building lattices. *Information processing letters*, 71(5) :199–204, 1999. p 41

[144] Sergei A. Obiedkov and Vincent Duquenne. Attribute-incremental construction of the canonical implication basis. *Annals of Mathematics and Artificial Intelligence*, 49(1-4) :77–99, April 2007. p 39, p 119

[145] Oystein Öre et al. Theory of equivalence relations. *Duke Mathematical Journal*, 9(3) :573–627, 1942. p 20

[146] Oystein Öre et al. Some studies on closure relations. *Duke math. J*, 10 :761–785, 1943. p 28

[147] Suhail Owais, Petr Gajdoš, and Václav Snášel. Usage of genetic algorithm for lattice drawing. In *CEUR Workshop Proceedings*, volume 162, 2005. p 42

[148] John L Pfaltz. Representing numeric values in concept lattices. In *CLA*, 2007. p 29

[149] Geraldine Polaillon. *Organisation et interprétation par les treillis de Galois de données de type multivalué, intervalle ou histogramme.* PhD thesis, Université de Paris Dauphine, 1998. p 44

[150] Uta Priss. Description logic and faceted knowledge representation. In *Proceedings of the 1999 International Workshop on Description Logics (DL'99), Linköping, Sweden, July 30 - August 1, 1999*, 1999. p 42, p 120

[151] Uta Priss. Faceted knowledge representation. *Electron. Trans. Artif. Intell.*, 4(C) :21–33, 2000. p 42

[152] Uta Priss. Lattice-based information retrieval. *Knowledge Organization*, 27(3) :132–142, 2000. p 42, p 45, p 46

[153] Uta Priss. Formal concept analysis in information science. *ARIST*, 40(1) :521–543, 2006. p 17

[154] Uta Priss. Fca software interoperability. In *Proceedings of the Sixth International Conference on Concept Lattices and Their Applications (CLA 2008)*, pages 133–144, 2008. p 121

[155] Uta Priss. Relation algebra operations on formal contexts. In Sebastian Rudolph, Frithjof Dau, and Sergei O. Kuznetsov, editors, *Conceptual Structures : Leveraging Semantic Technologies*, volume 5662 of *Lecture Notes in Computer Science*, pages 257–269. Springer Berlin Heidelberg, 2009. p 119

[156] Heiko Reppe. *Three generalisations of lattice distributivity : an FCA perspective*. PhD thesis, TU Dresden, 2011. p 7

[157] Jose Manuel Rodriguez-Jimenez, Pablo Cordero, Manuel Enciso, and Angel Mora. A generalized framework to consider positive and negative attributes in formal concept analysis. In *Proceedings of the Eleventh International Conference on Concept Lattices and Their Applications, Košice, Slovakia, October 7-10, 2014.*, pages 267–278, 2014. p 39

[158] Jose Manuel Rodriguez-Jimenez, Pablo Cordero, Manuel Enciso, and Angel Mora. Negative attributes and implications in formal concept analysis. *Procedia Computer Science*, 31(0) :758 – 765, 2014. 2nd International Conference on Information Technology and Quantitative Management, {ITQM} 2014. p 29, p 39

[159] Estrella Rodríguez-Lorenzo, Karell Bertet, Pablo Cordero, Manuel Enciso, and Angel Mora. The direct-optimal basis via reductions. In *Proceedings of the Eleventh International Conference on Concept Lattices and Their Applications, Košice, Slovakia, October 7-10, 2014.*, pages 145–156, 2014. p 39

[160] Estrella Rodríguez-Lorenzo and Karell Bertet. From implicational systems to direct-optimal bases : A logic-based approach. *Applied Mathematics & Information Sciences*, 9(305) :305–317, 2015. p 38, p 39

[161] Mohamed Hacene Rouane, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. A proposal for combining formal concept analysis and description logics for mining relational data. In *Proceedings of the 5th International Conference on Formal Concept Analysis*, ICFCA'07, pages 51–65, Berlin, Heidelberg, 2007. Springer-Verlag. p 43, p 45

[162] Uwe Ryssel, Felix Distel, and Daniel Borchmann. Fast algorithms for implication bases and attribute exploration using proper premises. *Annals of Mathematics and Artificial Intelligence*, 70(1-2) :25–53, 2014. p 39

[163] Václav Snásel, Zdenek Horak, Jana Kocibova, and Ajith Abraham. Analyzing social networks using FCA : complexity aspects. In *Proceedings of the 2009 IEEE/WIC/ACM International Conference on Web Intelligence and International Conference on Intelligent Agent Technology - Workshops, Milan, Italy, 15-18 September 2009*, pages 38–41, 2009. p 45

[164] Gregor Snelting. Concept lattices in software analysis. In Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors, *Formal Concept Analysis*, volume 3626 of *Lecture Notes in Computer Science*, pages 272–287. Springer Berlin Heidelberg, 2005. p 119

[165] Gerd Stumme. Off to new shores : Conceptual knowledge discovery and processing. *Int. J. Hum.-Comput. Stud.*, 59(3) :287–325, September 2003. p 45

[166] Gerd Stumme and Alexander Maedche. FCA-Merge : Bottom-up merging of ontologies. In B. Nebel, editor, *Proc. 17th Intl. Conf. on Artificial Intelligence (IJCAI '01)*, pages 225–230, 2001. p 120

[167] Gerd Stumme and Rudolf Wille. A geometrical heuristic for drawing concept lattices. In Roberto Tamassia and Ioannis G. Tollis, editors, *Graph Drawing*, volume 894 of *Lecture Notes in Computer Science*, pages 452–459. Springer Berlin Heidelberg, 1995. p 42

[168] Thomas Tilley. Tool support for fca. In Peter Eklund, editor, *Concept Lattices*, volume 2961 of *Lecture Notes in Computer Science*, pages 104–111. Springer Berlin Heidelberg, 2004. p 130

[169] Elena Tsiporkova, Veselka Boeva, and Elena Kostadinova. Mapreduce and fca approach for clustering of multiple-experiment data compendium. In Patrick De Causmaecker, Joris Maervoet, Tommy Messelis, Katja Verbeeck, and Tim Vermeulen, editors, *The 23rd Benelux Conference on Artificial Intelligence*. BNAIC 2011, 2011. p 78

[170] Petko Valtchev and Vincent Duquenne. Towards scalable divide-and-conquer methods for computing concepts and implications. In *Proceedings of the 4th Intl. Conference Journées de l'Informatique Messine (JIM'03) : Knowledge Discovery and Discrete Mathematics, Metz (FR)*, pages 3–6, 2003. p 39

[171] Petko Valtchev and Vincent Duquenne. On the merge of factor canonical bases. In Raoul Medina and Sergei Obiedkov, editors, *Formal Concept Analysis*, volume 4933 of *Lecture Notes in Computer Science*, pages 182–198. Springer Berlin Heidelberg, 2008. p 39, p 119

[172] Petko Valtchev and Rokia Missaoui. Building concept (galois) lattices from parts : Generalizing the incremental methods. In Harry S. Delugach and Gerd Stumme, editors, *Conceptual Structures : Broadening the Base*, volume 2120 of *Lecture Notes in Computer Science*, pages 290–303. Springer Berlin Heidelberg, 2001. p 41

[173] Petko Valtchev, Rokia Missaoui, and Robert Godin. Formal concept analysis for knowledge discovery and data mining : The new challenges. In Peter Eklund, editor, *Concept Lattices*, volume 2961 of *Lecture Notes in Computer Science*, pages 352–371. Springer Berlin Heidelberg, 2004. p 5, p 39

[174] Petko Valtchev, Rokia Missaoui, and Pierre Lebrun. A partition-based approach towards constructing galois (concept) lattices. *Discrete Math.*, 256(3) :801–829, October 2002. p 41, p 119

[175] Jean-François Viaud, Karell Bertet, Christophe Demko, and Rokia Missaoui. The reverse doubling construction. In *KDIR 2015 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, part of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015), Volume 1, Lisbon, Portugal, November 12-14, 2015*, pages 350–357, 2015. p 7

[176] Jean-François Viaud, Karell Bertet, Christophe Demko, and Rokia Missaoui. Subdirect decomposition of contexts into subdirectly irreducible factors. In *Proceedings of the International Workshop on Formal Concept Analysis and Applications, FCA&A 2015, co-located with 13th International Conference on Formal Concept Analysis (ICFCA 2015), Nerja, Málaga, Spain, June 23-26, 2015.*, pages 49–64, 2015. p 7

[177] Muriel Visani, Karell Bertet, and Jean-Marc Ogier. Navigala : an Original Symbol Classifier Based on Navigation through a Galois Lattice. *International Journal on Pattern Recognition and Artificial Intelligence (IJPRAI)*, 2011. p 8, p 42

[178] Tom White. *Hadoop : The definitive guide.* " O'Reilly Media, Inc.", 2012. p 77

[179] Rudolf Wille. Subdirekte produkte und konjunkte summen. *Journal für die reine und angewandte Mathematik*, 0239_0240 :333–338, 1969. p 7

[180] Rudolf Wille. Subdirekte Produkte vollständiger Verbände. *J. reine angew. Math.*, 283/284 :53–70, 1976. p 7

[181] Rudolf Wille. Restructuring lattice theory : an approach based on hierarchies of concepts. In Ivan Rival, editor, *Ordered sets*, pages 445–470. Reidel, 1982. p 5, p 126

[182] Rudolf Wille. Subdirect decomposition of concept lattices. *Algebra Universalis*, 17 :275–287, 1983. p 7

[183] Rudolf Wille. Sur la fusion des contextes individuels. *Mathématiques et Sciences Humaines*, 85 :57–71, 1984. p 43

[184] Rudolf Wille. Subdirect product construction of concept lattices. *Discrete Mathematics*, 63(2-3) :305–313, 1987. p 7

[185] Rudolf Wille. Geometric representation of concept lattices. In Otto Optiz, editor, *Conceptual and Numerical Analysis of Data*, pages 239–255. Springer Berlin Heidelberg, 1989. p 42

[186] Rudolf Wille. Concept lattices and conceptual knowledge systems. *Computers & Mathematics with Applications*, 23(6) :493 – 515, 1992. p 130

[187] Rudolf Wille. Conceptual structures of multicontexts. In Peter W. Eklund, Gerard Ellis, and Graham Mann, editors, *Conceptual Structures : Knowledge Representation as Interlingua*, volume 1115 of *Lecture Notes in Computer Science*, pages 23–39. Springer Berlin Heidelberg, 1996. p 43

[188] Rudolf Wille. Formal concept analysis as mathematical theory of concepts and concept hierarchies. In Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors, *Formal Concept Analysis*, volume 3626 of *Lecture Notes in Computer Science*, pages 1–33. Springer Berlin Heidelberg, 2005. p 15

[189] Rudolf Wille. Human being and mathematics, logical and mathematical thinking. In Sebastian Rudolph, Frithjof Dau, and Sergei O. Kuznetsov, editors, *Conceptual Structures : Leveraging Semantic Technologies*, volume 5662 of *Lecture Notes in Computer Science*, pages 66–85. Springer Berlin Heidelberg, 2009. p 5, p 17

[190] Biao Xu, Ruairi de Fréin, Eric Robson, and Micheal Ó Foghlú. Distributed formal concept analysis algorithms based on an iterative mapreduce framework. In Florent Domenach, Dmitry I. Ignatov, and Jonas Poelmans, editors, *Formal Concept Analysis*, volume 7278 of *Lecture Notes in Computer Science*, pages 292–308. Springer Berlin Heidelberg, 2012. p 78, p 90

[191] Xifeng Yan and Jiawei Han. Closegraph : Mining closed frequent graph patterns. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 286–295, New York, NY, USA, 2003. ACM. p 43

[192] Serhiy A Yevtushenko. System of data analysis "concept explorer". In *Proceedings of the 7th national conference on Artificial Intelligence KII*, volume 2000, 2000. p 93, p 131, p 152

Table des figures

Liste des tableaux

Liste des Algorithmes

Index