HABILITATION A DIRIGER
DES RECHERCHES

EN INFORMATIQUE

présentée à

l'Université de La Rochelle

par

KARELL BERTET

Structure de treillis

Contributions structurelles et algorithmiques
Quelques usages pour des données images

soutenue le 14 Juin 2011 devant le jury composé de

Pr. Bernard Monjardet	Université Paris 1	Rapporteur
Pr. Amedeo Napoli	DR CNRS, Loria, Nancy	Rapporteur
Pr. Karl Trombe	Ecole des Mines, Loria, Nancy	Rapporteur
Pr. Engelbert Mephu Nguifo	Université B. Pascal, Clermont-Ferrand	
Pr. Lhouari Nourine	Université B. Pascal, Clermont-Ferrand	
Pr. Jean-Marc Ogier	Université de La Rochelle	
Pr. Luigi Santocanale	Université de Provence, Marseille	

A Nina, Théo et Laurent

Remerciements

Je souhaite tout d'abord adresser mes remerciements aux personnes qui me font l'honneur de faire partie de ce jury, à savoir Bernard Monjardet, Amedeo Napoli, Karl Trombe, Engelbert Mephu Nguifo, Lhouari Nourine, Jean-Marc Ogier et Luigi Santocanale. Je suis honorée de pouvoir leur présenter mon travail.

Je remercie également toutes les personnes avec qui j'ai eu l'occasion de collaborer depuis le début de ma carrière, collaborations qui m'ont beaucoup apporté tant d'un point de vue scientifique que d'un point de vue personnel. Je citerai en particulier Michel Morvan qui m'a fait découvrir et aimer le monde la recherche pendant mes années de thèse, Jean-Marc Ogier qui m'a fait découvrir le domaine de l'image à mon arrivée à La Rochelle, Muriel Visani avec qui je partage mon bureau dans une ambiance amicale et conviviale, et qui m'a initiée à la fouille de données, Alain Bouju et Georges Louis pour les échanges fructueux que nous avons pu avoir cette dernière année autour de la représentation des connaissances. Je tiens également à remercier les membres de la communauté francophone des treillis grâce à qui j'ai pu maintenir une activité scientifique de qualité autour de la structure de treillis.

Je souhaite également remercier les étudiants qui ont contribué à l'élaboration de ce travail : Mickaël Coustaty, Nathalie Girard, Marçal Rossinol, Simon Bernard, Antoine Mercier, Philippe Sachot, Romain Bertrand, Thang Le Ngoc, Hamza Khene et Mustapha Al-Haj. Je remercie tout particulièrement Stéphanie Guillas, première étudiante en thèse que j'ai encadrée, et avec qui j'ai eu des échanges fructueux et amicaux.

Je remercie également le directeur du laboratoire, Remy Mullot pour la confiance qu'il a pu m'accorder sur un plan scientifique, ainsi que Michel Ménard et Arnaud Revel pour les conseils qu'ils ont pu m'apporter ces dernièrs mois autour de la préparation de cette HdR.

Pour finir, une pensée toute particulière pour Théo, Nina et Laurent à qui je dédie ce mémoire.

Résumé

Ce document présente une synthèse de mes travaux de recherche depuis la fin de ma thèse. Ces travaux portent sur la structure de treillis, avec des contributions qui se situent à la fois sur un plan fondamental, avec des aspects structurels et algorithmiques des treillis, et sur un plan applicatif, avec l'étude de quelques usages de treillis pour des données images. Ainsi, mes travaux de recherche s'organisent naturellement en deux parties distinctes.

La première partie a pour objectif de présenter les concepts de base de la théorie des treillis et les principaux algorithmes de génération des objets qui la composent, de façon suffisament précise et complète pour pouvoir être exploitée dans un cadre applicatif. Les algorithmes sont détaillés, les notions clé illustrées par des exemples. Mes contributions, initiées dans le cadre de mes travaux de thèse, portent sur la définition ordinale d'un treillis. J'ai en particulier proposé divers algorithmes, et identifié la base canonique directe d'un treillis. Ces travaux ont donné lieu à une bibliothèque de génération et de manipulation de ces objets.

La seconde partie est consacrée à quelques usages des treillis en informatique pour traiter des données, et en particulier des images. Mes contributions concernent à la fois l'exploration ou la fouille de données dans un objectif de classification, et la représentation de relations spatiales entre primitives issues d'une image, en particulier par une ontologie. Ces travaux ont donné lieu à un logiciel de reconnaissance d'images de symboles par navigation dans un treillis. Un travail conséquent de synthèse des méthodes à base de treillis existant en fouille de données et en représentation ontologique des connaissances est présenté, l'objectif étant de positonner les différentes approches les unes par rapport aux autres.

Avant-propos

Contexte initial

Ce document présente une synthèse de mes travaux de recherche menés au laboratoire Informatique, Image et Interaction - L3I EA 2118 - de l'université de La Rochelle depuis 1999, après avoir soutenu un doctorat en algorithmique à l'université Paris 7 en 1998.

Mes travaux de thèse se positionnent sur un plan fondamental autour d'aspects structurels et algorithmiques de la structure de treillis. Bien que les champs applicatifs possibles de la structure de treillis en informatique soient divers et variés, je ne les ai pas étudiés pendant mes travaux de thèse. C'est pourquoi j'ai naturellement voulu investiguer dans cette direction à mon arrivée au laboratoire L3I. Mes travaux se sont alors orientés vers une *utilisation de la structure de treillis dans un objectif de reconnaissance d'images* (**partie II**), et plus particulièrement d'images graphiques. En effet, la structure de treillis via le treillis des concepts est naturellement liée à la problématique de classification de par le regroupement hiérarchique des objets qu'elle peut induire. Je me suis approprié les problématiques propres aux domaines du traitement d'image et de la classification, domaines nouveaux pour moi. Cette appropriation a été grandement facilitée grâce à Jean-Marc Ogier qui m'a proposé une collaboration scientifique dans des projets autour d'images de documents qu'il gérait (Madonne 2004-2006, Navidomass 2006-2009, Epeires 2005-2006). Nous avons ainsi proposé une méthode originale `Navigala` de classification par navigation dans un treillis des concepts, travaux réalisés dans le cadre de la thèse de Stéphanie Guillas (2004-2007). Mon implication au sein du laboratoire L3I s'est ensuite vu renforcée par la prise de *responsabilité de l'équipe Imédoc* (Images, Média et Documents Numériques) en 2007, puis de l'équipe IDDC (Images, Documents et Données Complexes) en 2011.

En parallèle, j'ai maintenu une activité plus fondamentale autour d'*aspects structurels et algorithmiques des treillis* (**partie I**), et en particulier de la base canonique directe d'un treillis. Ces travaux font suite aux propriétés du graphe de dépendance

d'un treillis identifiées pendant mes travaux de thèse, et m'ont amenée à m'intéresser plus particulièrement à la notion de système de fermeture, notion centrale de la théorie des treillis, autour de laquelle gravitent les structures de treillis, treillis des concepts, treillis de Galois, table binaire ou encore système de règles. J'ai ainsi pu constater le peu d'outils informatiques existants pour manipuler un système de fermeture. C'est pourquoi j'ai réalisé une bibliothèque `lattice` dont le but est d'exploiter les liens bijectifs entre ces objets. Dans le même temps, j'ai participé à la structuration de la *communauté francophone des treillis* en 2007, communauté qui se retrouve autour de la structure de treillis.

Le tableau suivant présente une synthèse des résultats obtenus pour chacune de ces deux activités.

	Publications		Encadrement		Développement
	Revues	Conf.	Thèses	Stages	
(**Partie II**)					
Usages en image	5	9	2	8	logiciel NAVIGALA
(**Partie I**)					
Structure et algorithmes	2+3(nat.)	9	0	0	bibliothèque LATTICE

Motivation et démarche

Il existe plusieurs façons de distinguer la recherche fondamentale de la recherche appliquée. En informatique, une distinction peut s'établir en considérant les approches et résultats - à savoir approche par preuve ou par expérimentation. Une autre distinction met en avant les motivations même de la recherche. Alors qu'une recherche fondamentale cherche avant tout à *comprendre*, une recherche plus appliquée vise à *construire* de nouvelles approches. Bien entendu, recherche fondamentale et recherche appliquée s'enrichissent mutuellement, et l'algorithmique se positionne alors comme un élément central.

En tant qu'algorithmicienne, ma motivation principale repose sur cet *enrichissement mutuel autour de la structure de treillis*, à la fois en terme de contribution, mais également de diffusion. J'ai ainsi fait le choix d'appréhender la structure de treillis selon ce point de vue, en maintenant en parallèle une activité fondamentale et une application à l'image. Je citerai deux exemples illustratifs :

– *Enrichissement de l'application par le fondamental :* les treillis - de Galois ou de

fermés - générés à partir de données sont rarement utilisés dans leur globalité par les méthodes qui les exploitent. Ainsi, leur génération, coûteuse à la fois en temps et en mémoire, n'est pas toujours indispensable. Une connaissance approfondie des propriétés structurelles et algorithmiques des treillis permet alors de caractériser, voire de ne générer, que les éléments utilisés - éléments irréductibles, ou encore successeurs immédiats d'un concept par exemple. La méthode de classification par navigation que nous avons proposée [34] exploite cette possibilité en proposant une *génération à la demande* des successeurs immédiats d'un concept, d'où un gain considérable en temps de calcul lors de la phase d'apprentissage.

– *Enrichissement du fondamental par l'applicatif :* j'ai par exemple pu observer que les treillis de Galois obtenus après discrétisation d'un ensemble de données numériques s'avèrent posséder deux propriétés structurelles importantes - la co-atomisticité et la $\vee$-complémentarité - et forment ainsi une nouvelle classe de treillis que nous avons nommée la classe des *treillis dichotomiques*. Une étude structurelle de cette classe de treillis pourrait permettre de proposer un jeu algorithmique adapté. Nous avons des contributions en ce sens [18].

Mes travaux de recherche se situent à la fois sur un plan fondamental - avec des contributions d'ordre théorique où des propriétés et algorithmes sont établis à l'aide de preuves - mais également sur un plan applicatif - avec des propositions de nouvelles méthodes validées par des études expérimentales. Ce positionnement trouve écho au niveau national à travers la communauté francophone des treillis qui regroupe des équipes aussi bien en mathématiques discrètes, informatique fondamentale, ou encore informatique appliquée, et plus particulièrement en intelligence artificielle.

Cet angle d'approche permet également de dégager des *liens entre les différentes approches* qui peuvent être issues de divers domaines, permettant de ne pas redécouvrir les mêmes résultats sous des formalismes différents. Ainsi, il n'est pas surprenant de retrouver les mêmes notions, mais aussi les mêmes résultats et algorithmes sous des terminologies différentes. A l'inverse, on peut également trouver des résultats originaux connus seulement dans un domaine bien spécifique, alors qu'ils pourraient s'étendre facilement aux autres domaines. C'est pourquoi il serait profitable de créer des liens entre ces différents domaines qui manipulent les mêmes outils.

A nouveau je citerai deux exemples illustratifs. Dans nos travaux autour de la base canonique directe, nous avons montré qu'elle a été redécouverte au moins cinq fois sous des formes différentes, en particulier dans le domaine de la théorie des treillis, de la fouille de données, ou encore en logique propositionnelle. L'équivalence entre ces différentes bases permet d'unifier les différentes approches, avec en particulier la mise en valeur du lien qui unit la base canonique directe à la relation de dépendance d'un treillis.

Concernant l'utilisation du treillis de Galois en classification, des points communs avec les algorithmes génétiques, les méthodes à noyaux (variante des SVM) ou encore les réseaux de neurones ont été observés dans la littérature. Nos travaux se sont intéressés plus particulièrement aux liens qui l'unissent aux représentations hiérarchiques arborescentes.

L'applicabilité d'une recherche fondamentale n'est pas chose aisée. Elle demande des compétences issues de plusieurs domaines, ainsi qu'une volonté réelle de mise en pratique, c'est pourquoi elle n'est souvent mentionnée qu'en introduction des articles du domaine. Le *domaine de l'image, thème de recherche principal du laboratoire L3I*, se prête particulièrement bien à cette démarche. La *diversité du champ applicatif* de l'image, ainsi que la *richesse des données* que l'on peut en extraire, ouvre la voie à de nombreuses applications. La diversité des projets portés par le laboratoire l'atteste.

J'ai rapidement fait le choix d'*exploiter la structure de treillis des concepts* introduite dans le cadre de l'analyse formelle des concepts (AFC) en pleine émergence depuis quelques années en informatique, émergence qui s'explique à la fois par la part grandissante de l'informatique dans la plupart des champs disciplinaires, ce qui conduit à la production de données en quantité de plus en plus importante ; mais également par la récente montée en puissance des ordinateurs qui, bien que la taille du treillis puisse être exponentielle en fonction des données dans le pire des cas, rend possible le développement d'un grand nombre d'applications le concernant. La taille du treillis reste cependant raisonnable en pratique et la nécessité d'algorithmes efficaces pour manipuler ces structures est un défi majeur.

Dans le cas particulier de données images et plus particulièrement d'images de documents, nous avons choisi d'investiguer les *problématiques de fouille de données, et de représentation des connaissances*, problématiques naturellement liées à la notion de concept du treillis des concepts - un concept est un regroupement maximal d'objets possédant des caractéristiques communes. Une attention particulière a été portée à un positionnement complémentaire de travaux réalisés au sein du laboratoire qui reposent sur des compétences issues de domaines variés, alliant analyse d'images, fouille de données, et représentation des connaissances. Le travail réalisé autour de données image, et en particulier des images graphiques issues de documents, est avant tout un travail d'équipe. La bibliothèque `lattice` a ainsi été développée pour un meilleur usage des algorithmes utilisant des treillis au sein du laboratoire.

Si je devais faire ressortir un fil conducteur à ces travaux, ce serait l'*intégration progressive de données complexes* décrivant l'image, tout en garantissant la lisibilité de la structure. En effet, la signature d'une image, qui porte une information condensée

plus ou moins complexe qui permet de la décrire mais aussi de la caractériser, peut s'organiser sous forme d'un vecteur numérique décrivant l'organisation des pixels dans l'image, mais également sous une forme plus complexe décrivant l'agencement topologique de primitives extraites de l'image (traits, arcs de cercles, ou plus généralement régions de l'image). Par ailleurs, les images de documents sont des images issues d'un domaine souvent porteur d'une connaissance métier forte (lettrines, symboles, plan cadastral) qu'il semble naturel d'intégrer aux données pour une meilleure représentation de l'information. Ce positionnement trouve écho au sein du laboratoire, il a en effet été identifié comme un nouveau verrou scientifique à aborder au cours des prochaines années.

Apports

Nous présentons ici nos résultats scientifiques majeurs :

- **Méthode de reconnaissance d'image par navigation dans un treillis des concept.** [34] Nommée Navigala (NAVIgation into GAlois LAttice), cette méthode est dédiée à la reconnaissance d'objets graphiques détériorés, et plus particulièrement d'images de symboles. Son originalité repose sur le principe de navigation réalisée à partir du concept minimal, jusqu'à atteindre un concept final représentatif d'une classe. Le verrou principal que nous avons levé ici est une extension de la navigation proposée dans un arbre de décision à une navigation dans un treillis.
- **Etude des treillis dichotomiques.** [18] Les treillis générés par la méthode Navigala, et plus généralement à partir d'une signature numérique discrétisée, forment une classe de treillis intermédiaires entre les modèles hiérarchiques d'arbres et les treillis de manière générale possédant des propriétés structurelles fortes. Nous avons ainsi mis en avant un lien de fusion entre treillis et arbres. Nous avons également exploité les propriétés structurelles des treillis dichotomiques pour proposer un algorithme de discrétisation locale efficace.
- **Représentation de l'information spatiale entre primitives d'une image.** [19] Nous avons ici exploré les deux approches standard de manipulation de l'information spatiale entre les primitives d'une image. La première approche se positionne dans un objectif de reconnaissance d'images de symboles, une signature structurelle est ainsi proposée : il s'agit d'un sac de chemins du graphe des relations spatiales entre primitives. La seconde approche exploite les relations spatiales entre primitives issues d'images de lettrines, selon une stratégie de représentation ontologique des connaissances.

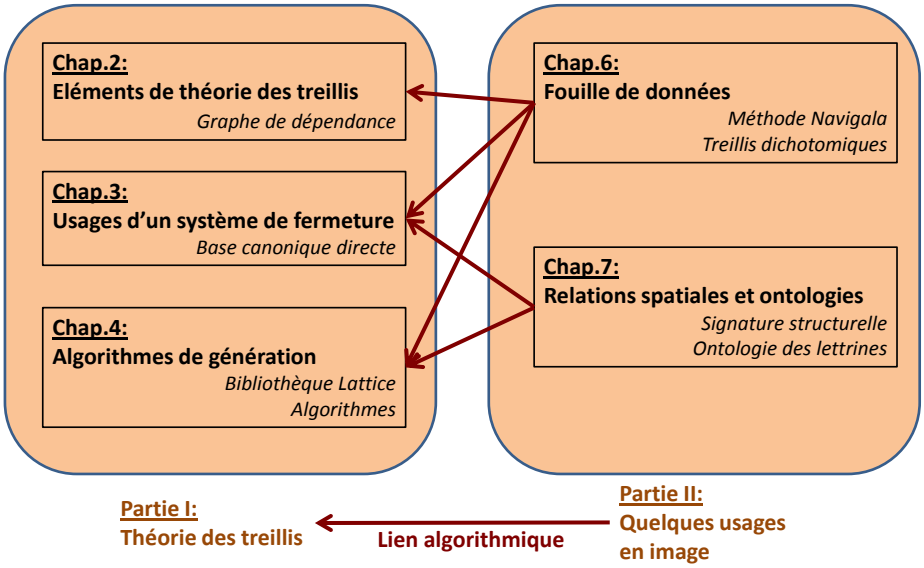


FIGURE 1 – Organisation du mémoire, avec les principales contributions en italique.

- **Etude de la base canonique directe d'un treillis.** [16][11] Introduite à l'aide d'un algorithme de génération, ou encore comme la base encodée par le graphe de dépendance d'un treillis, nous avons par la suite montré que la base canonique directe a été redécouverte au moins cinq fois sous des formes différentes. L'équivalence entre ces différentes bases permet d'unifier les différentes approches, avec en particulier la mise en valeur du lien qui unit la base canonique directe à la relation de dépendance d'un treillis.
- **Bibliothèque de manipulation d'un système de fermeture.** Nommée LATTICE, cette bibliothèque développée en Java définit un système de fermeture comme élément principal, instanciable aussi bien par un système implicatif que par un contexte. Les liens bijectifs entre les différents composants de la théorie des treillis sont ainsi exploités, pour une manipulation efficace dans un contexte pédagogique ou applicatif expérimental. Par ailleurs, cette biblitohèque intègre de nouveaux algorithmes que nous avons développés. Elle est disponible sous licence LGPL.

Organisation de ce mémoire

Ce manuscrit comprend deux parties distinctes, résumées par la figure 1 où les principales contributions sont précisées en italique. L'introduction à chacune de ces deux parties décrit plus en détails la démarche et les contributions.

Partie I : Structure de treillis : concepts de base et algorithmes. La première partie ne se structure pas autour des contributions, mais s'organise de façon didactique comme un cours, les contributions étant mentionnées au fil des sections, et précisées en italique,. L'objectif est de présenter les concepts de base et les principaux algorithmes de génération de la théorie des treillis de façon suffisament précise et complète pour pouvoir être exploitée dans un cadre applicatif. Les algorithmes sont détaillés, les notions clé illustrées par des exemples. Nos contributions dans ce domaine, initiées dans le cadre de nos travaux de thèse, portent sur la définition ordinale d'un treillis, ainsi que la base canonique directe qui le représente.

Partie II : Quelques usages pour des données images. Cette seconde partie est consacrée à quelques usages des treillis en informatique pour traiter des données, et en particulier des images. Nos travaux ont porté à la fois sur l'exploration ou la fouille de données dans un objectif de classification, et sur la représentation de relations spatiales entre primitives issues d'une image, en particulier par une ontologie. Un travail conséquent de synthèse des méthodes issues de l'AFC existant en fouille de données et en représentation ontologique des connaissances est présenté, l'objectif étant de positionner les différentes approches les unes par rapport aux autres.

Table des matières

Première partie

Structure de treillis : concepts de base et algorithmes

Chapitre 1

Introduction

Le premier ouvrage de référence de la théorie des treillis est la première édition du livre de Birkhoff en 1940 [47]. Cependant, la notion de treillis a été introduite dès la fin du $19^{\text{ème}}$ siècle comme une structure algébrique munie de deux opérateurs appelés borne inférieure et borne supérieure, ou encore comme une structure ordonnée, définie par l'existence d'éléments particuliers appelés borne supérieure et inférieure.

Un résultat fondamental de la théorie des treillis se constitue autour d'un résultat principal qui établit que tout treillis fini est isomorphe au *treillis de Galois* ou *treillis des concepts* de sa *table* [45]. La table d'un treillis est une table binaire constituée des *éléments irréductibles* du treillis. Il s'agit d'éléments qui ne sont pas une borne supérieure ou une borne inférieure d'une partie ne les contenant pas. Plus précisemment, la table est composée en colonne des sup-irréductibles et en ligne des inf-irréductibles. Elle fournit ainsi une représentation condensée du treillis, à partir de laquelle sa reconstruction est possible via le treillis des concepts. Défini plus généralement à partir de données décrites par une table binaire objet x attribut, nommée *contexte*, le *treillis de concepts* est un graphe possédant la propriété de treillis. Les noeuds du graphe sont des concepts reliés par intension/extension, où un concept est un sous-ensemble maximal d'objets présentant des attributs communs.

La notion de fermeture, pilier de ce premier résultat, est utilisée explicitement dans une deuxième bijection établissant que tout treillis est isomorphe au *treillis de fermés* d'une base de règles d'implication le représentant. Il s'agit de règles du type "si" "alors" qui fournissent une représentation du treillis sur la base de ses seuls éléments inf-irréductibles, à partir de laquelle sa reconstruction est possible. Parmi les systèmes de règles d'implication équivalents par leur représentativité d'un même treillis, la *base canonique* et la *base canonique directe* se distinguent. Défini plus généralement à partir d'un *système de fermeture* - ensemble muni d'un opérateur de fermeture - le treillis de fermés est un graphe dont les noeuds sont les parties de S stables par fermeture, et reliées par inclusion.

L'existence de liens bijectifs entre treillis, contextes - via la *table binaire* - et systèmes d'implication - via la *base canonique* ou la *base canonique directe* - est une conséquence

directe de ces deux résultats fondamentaux :

1. Tout treillis est le treillis des concepts de sa table binaire.
2. Tout treillis est le treillis de fermés de sa base canonique et de sa base canonique directe.

Dans un cadre applicatif, la structure de treillis des concepts ou de treillis de fermés fournit une représentation très intuitive des données lorsqu'elles sont décrites par une table binaire ou bien des règles d'implication. Depuis les années 2000, l'émergence de l'analyse de concepts formels (FCA) [58] dans divers domaines de l'informatique, que ce soit en extraction et représentation des connaissances, dans les domaines des ontologies ou encore des bases de données, a ainsi mis en avant les structures de treillis des concepts et de treillis de fermés. L'émergence de cette structure de treillis s'explique à la fois par la part grandissante de l'informatique dans la plupart des champs disciplinaires, ce qui conduit à la production de données en quantité de plus en plus importante ; mais également par la montée en puissance des ordinateurs qui, bien que la taille du treillis puisse être exponentielle en fonction des données dans le pire des cas, rend possible le développement d'un grand nombre d'applications le concernant. La taille du treillis reste cependant raisonnable en pratique, et la nécessité d'algorithmes efficaces pour manipuler ces structures est un défi majeur.

Les retombées structurelles et algorithmiques des liens bijectifs entre ces trois structures de treillis, tables binaire et base de règles sont importantes. Par exemple, les seuls concepts irréductibles du treillis des concepts portent une information "atomique" pertinente et polynomiale en la taille du contexte, concepts qu'il est possible de générer en évitant de construire le treillis dans sa totalité car sa taille est exponentielle dans le pire des cas en celle du contexte. Citons également les propriétés des bases canoniques d'un treillis exploitées pour extraire une représentation de données sous forme de règles d'implication à la fois canonique, non redondante et complète. Ou encore la possibilité de "réduire" un système de règles quelconques en ne conservant que certains éléments irréductibles qui le caractérisent. Dans un cadre applicatif, une exploitation efficace du jeu algorithmique existant pour manipuler ces structures nécessite cependant une bonne connaissance du formalisme et des propriétés structurelles de la théorie des treillis afin d'identifier les outils algorithmiques adaptés pour un problème donné.

Notre approche diffère cependant de l'approche classique proposée par l'AFC. Nous proposons comme substitut la mise en avant d'un système de fermeture - qui peut être un contexte, ou encore un système de règles d'implication - à partir duquel se définit naturellement un treillis de fermés. Le treillis des concepts s'en déduit. Comme nous essaierons de le montrer à travers cette partie, cet angle d'approche permet de contribuer à une généricité théorique, mais surtout algorithmique.

Le chapitre 2 pose les principales définitions issues de la théorie des treillis. Le chapitre 3 introduit les structures de treillis des concepts et de treillis de fermés présentées comme deux usages classiques d'un système de fermeture. Le chapitre 4 s'intéresse aux

aspects algorithmiques permettant de générer ces objets.

Chapitre 2 : Eléments de théorie des treillis. Le chapitre 2 introduit les deux définitions fondamentales d'un treillis que sont la définition algébrique et la définition ordinale. Nous ne traiterons dans ce document que des treillis dans le cas fini. La théorie des treillis utilise l'existence d'éléments particuliers d'un treillis que sont ses éléments *irréductibles* : alors que tout élément réductible d'un treillis peut s'obtenir à partir d'autres éléments par une opération de borne supérieure ou inférieure, les éléments irréductibles - éléments qui ne sont pas une borne supérieure ou une borne inférieure d'une partie ne les contenant pas - décrivent la structure même du treillis. C'est sur la base des irréductibles que se définissent la *table* et le *graphe de dépendance* d'un treillis, deux représentations distinctes d'un treillis qui contiennent l'information nécessaire à sa reconstruction.

Contributions. *Mes travaux de thèse se positionnent principalement autour de la définition ordinale d'un treillis, avec des résultats structurels et algorithmiques : construction du treillis des antichaînes maximales d'un ordre, construction similaire à celle du treillis des concepts ; calcul du plus petit treillis contenant un ordre donné [15] ; caractérisation et reconnaissance des treillis générés par duplications [5, 6] ; caractérisation de sous-treillis d'un treillis [12]. On peut également citer quelques propositions algorithmiques où les propriétés ordinales du treillis sont utilisées pour calculer les extensions faibles d'un ordre [13, 14], ou encore le pléthysme de certaines fonctions quasi-symétriques [10].*

Introduit également pendant mes travaux de thèse [2] sous une forme similaire, le graphe de dépendance est une représentation par les seuls inf-irréductibles définis à partir de la relation de dépendance *[68, 70] et des* générateurs minimaux *du treillis qui en valuent les arcs. La question de son utilité applicative, question posée lors de l'introduction de ce graphe, a trouvé récemment une réponse plus que satisfaisante de par les liens qui l'unissent à la base canonique directe.*

Tout treillis peut également se représenter sous forme ensembliste par une *famille de Moore*, famille de parties d'un ensemble stable par intersection, et contenant l'ensemble lui-même. L'apport principal de la représentation ensembliste réside dans le lien unissant treillis et système de fermeture, permettant ainsi d'étendre le champ applicatif des treillis à celui très riche des systèmes de fermeture. En effet, puisque tout treillis est représentable par un treillis de fermés, il l'est également par un système de fermeture. On retrouve des opérateurs de fermeture dans de nombreux domaines, que ce soit en logique, en topologie, en bases de données, en combinatoire, ou encore en analyse de données. C'est pourquoi nous avons choisi de mettre en avant la notion de système de fermeture, que l'on retrouve à la fois dans un contexte ou dans un système implicatif. Le treillis de fermé est ainsi positionné comme élément central autour duquel gravite à la fois la structure de treillis, mais aussi celles de treillis des concepts, treillis de Galois,

et systèmes implicatifs.

Chapitre 3 : Deux usages classiques d'un système de fermeture. Le chapitre 3 pose les définitions d'un *contexte* - et de son treillis des concepts - et d'un *système d'implication* - et de son treillis de fermés - que nous introduisons comme deux cas d'usage des systèmes de fermeture.

Pour le contexte, premier cas d'usage, la *connexion de Galois* entre les objets et les attributs qui la définissent induit l'existence de deux opérateurs de fermeture, le premier défini sur l'espace des objets et le second sur l'espace des attributs. Ces deux systèmes de fermeture sont à l'origine des propriétés du *treillis des concepts* d'un contexte, utilisé en analyse de données comme espace de recherche sous-jacent à des données binaires. Le treillis des concepts correspond en effet à la combinaison de deux treillis de fermés, le premier défini sur l'ensemble des attributs, le second sur l'ensemble des objets.

La notion de *treillis des concepts* est souvent assimilée à celle de *treillis de Galois*. Alors que la première terminologie fait référence à l'analyse formelle des concepts qui introduit la définition d'un concept, la seconde terminologie met en avant les propriétés de la connexion entre les objets et les attributs d'un contexte. Au contraire, nous introduisons ici le treillis de Galois comme une généralisation du treillis des concepts à des données plus complexes pour lesquelles il existe une connexion de Galois. En effet, alors qu'un contexte fournit une description ensembliste des objets - un objet est décrit par l'ensemble des attributs qu'il possède - l'existence d'une connexion de Galois n'est, quant à elle, pas limitée à des espaces ensemblistes, d'où la possibilité d'une description des objets dans un autre espace de description.

Un *système implicatif* - second cas d'usage d'un système de fermeture - est un ensemble de règles d'implication défini sur un ensemble, et pour lequel un mécanisme d'inférence se définit naturellement par un opérateur de fermeture. Le treillis de fermés est alors composé de toutes les parties qui vérifient l'ensemble des règles, appelées des *fermés*. En analyse de données, une règle décrit l'association systématique de certains attributs à d'autres, et le treillis de fermés est composé de toutes les parties d'attributs vérifiant ces liens d'association. Dans le domaine de la logique, les règles d'implication sont classiquement associées à une formule logique composée de clauses de Horn sous forme ensembliste, le treillis de fermés est alors composé de tous les sous-ensembles de variables qui vérifient la formule. La *base canonique* [60] et la *base canonique directe* constituent deux systèmes implicatifs particuliers, équivalents par leur représentativité d'un même treillis de fermés.

Contributions. *J'ai tout particulièrement investigué autour de la base canonique directe qui s'avère être équivalente au graphe de dépendance introduit pendant mes travaux de thèse [11]. En collaboration avec Mirabelle Nebut, nous avons par la suite proposé un algorithme de génération [16]. Puis, en collaboration avec Bernard Monjardet, nous avons montré qu'elle a été redécouverte au moins cinq fois sous des formes dif-*

férentes [11]. L'équivalence entre ces différentes bases permet d'unifier les différentes approches, avec en particulier la mise en valeur du lien qui unit la base canonique directe à la relation de dépendance d'un treillis.

Chapitre 4 : Aspects algorithmiques. L'exploitation générique d'un système de fermeture nécessite un jeu algorithmique efficace autour de quatre principaux problèmes de génération que sont la génération du treillis de fermés, la génération des éléments irréductibles, la génération de la base canonique et de la base canonique directe. Ces problèmes appartiennent à la classe des problèmes de génération difficile définis par une entrée de taille n et un résultat dont la taille est bornée par 2^n. Des techniques d'analyse spécifiques dirigées par la sortie sont par conséquent nécessaires. Le chapitre 4 décrit les principaux algorithmes de résolution de ces problèmes de génération.

Contributions. *En terme de contributions, j'ai proposé un algorithme incrémental de génération des fermés [16], différentes extensions de l'algorithme de Bordat [8], un algorithme générique de génération des éléments irréductibles, un calcul incrémental de la base canonique directe [4], une génération polynomiale de la base canonique directe à partir du treillis de fermés ou encore un comparatif entre les deux bases canoniques pour une utilisation conjointe [9].*

Finalement, j'ai implémenté ces différents algorithmes dans une bibliothèque développée en Java où le système de fermeture est défini comme élément principal, instanciable aussi bien par un système implicatif que par un contexte. Les liens bijectifs entre ces différents composants de la théorie des treillis, illustrés par la figure 1.1, sont ainsi exploités, pour une manipulation efficace dans un contexte pédagogique ou applicatif expérimental. Il est ainsi possible d'obtenir la table à partir d'un système de règles, ou encore la base canonique directe à partir d'un treillis. Par ailleurs, une classe dédiée à la représentation ordinale d'un treillis en permet une manipulation structurelle fine. Cette bibliothèque se positionne en complément des outils existants de manipulation d'un treillis qui, plûtot orientés données, posent le contexte comme élément central à partir duquel la génération du treillis ou des règles est proposée.

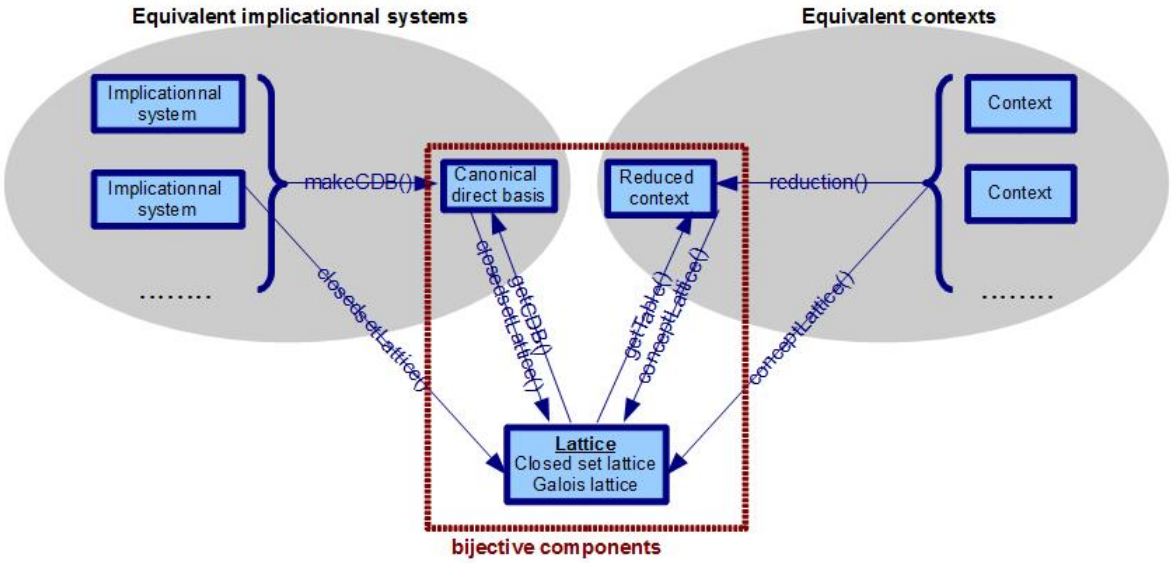


FIGURE 1.1 – Bijections

Chapitre 2

Eléments de théorie des treillis

Ce chapitre a pour but d'introduire les éléments de base de la théorie des treillis dans le cas fini, et s'organise en trois sections. La section 2.1 présente un bref historique de la théorie des treillis, à partir d'informations issues de [69]. La section 2.2 pose les définitions algébrique et ordinale d'un treillis qui introduisent toutes deux la notion de *borne supérieure* et de *borne inférieure* (section 2.2). La théorie des treillis repose en partie sur l'existence d'éléments particuliers d'un treillis que sont ses éléments irréductibles : alors que les éléments réductibles d'un treillis peuvent se reconstruire à partir d'autres éléments par un calcul de borne supérieure ou inférieure, les éléments irréductibles décrivent la structure même du treillis, et servent à en définir ses générateurs minimaux (section 2.2.2). C'est sur la base des irréductibles que se définissent des représentations d'un treillis qui contiennent l'information nécessaire à sa reconstruction. Sont introduits dans cette section la table d'un treillis ainsi que son graphe de dépendance (section 2.2.3).

La section 2.3 introduit la représentation d'un treillis sous forme ensembliste par une famille de parties d'un ensemble S, stable par intersection et contenant S lui-même. Cette représentation permet ainsi de manipuler de façon équivalente un treillis sous sa forme algébrique, ordinale ou ensembliste, car tout treillis est représentable par un treillis de fermés (section 2.3). Un treillis de fermés est classiquement associé à un *système de fermeture*, ensemble muni d'un *opérateur de fermeture*, et l'apport principal de cette représentation ensembliste réside dans le lien unissant treillis et système de fermeture (section 2.3.2). En effet, puisque tout treillis est représentable par un treillis de fermés, il l'est également par un système de fermeture.

2.1 Eléments historiques

1900 : Treillis algébrique et ordinal La notion de treillis définie comme une structure algébrique munie de deux opérateurs appelés borne inférieure et borne supérieure a été introduite dès la fin du $19^{\text{ème}}$ siècle par Dedekind - sous le terme de *dualgruppe* - et Schröder, puis oubliée. Elle a été redécouverte et développée au $20^{\text{ème}}$ siècle, sous

diverses formes et terminologies entre 1928 et 1936 dans les travaux de Menger, Klein, Stone, Birkhoff [46], Öre ou encore Von Neuman. L'introduction d'un treillis sous forme ordinale - structure ordonnée (i.e. transitive, antysymétrique et réflexive) définie par l'existence d'éléments particuliers appelés bornes supérieures et inférieures - trouve son origine dans les axiomatiques des treillis booléens (algèbres de Boole) dues par exemple à Pierce en 1880 [74], ou à Huntington [62]. Le terme de *treillis* a, quant à lui, été proposé par Birkoff lors du premier sympsosium sur la structure de treillis en 1938, pour être finalement repris dans son ouvrage de référence [47], alors que Menger utilisait le terme de *System von Dingen* et Öre celui de *structure*. S'ensuivent plusieurs conférences, ainsi que des publications, en particulier dans la revue *Algebra Universalis* créée en 1970, ainsi que dans la revue *Order*, revue du domaine créée en 1984. De nombreux ouvrages sur la théorie des treillis portent sur la définition ordinale, en particulier celui de Davey et Priestley [51] ou encore de Grätzer [59].

1970 : Treillis de Galois et treillis de fermés Un résultat fondamental de la théorie des treillis se constitue autour d'un résultat principal qui établit que tout treillis fini est isomorphe au *treillis de Galois* ou *treillis des concepts* de sa *table*[45]. Barbut et Monjardet introduisent ainsi le terme de *treillis de Galois*, alors que le terme de *correspondance de Galois* a quant à lui été introduit par Öre en 1944 dans [76]. Cette bijection peut se voir comme une conséquence des propriétés de fermeture portées par toute correspondance de Galois, et par conséquent s'applique également aux treillis de fermés [50], ainsi qu'à tout système muni d'un opérateur de fermeture. Plusieurs travaux font apparaître la notion d'éléments irréductibles d'un treillis qui permettent par exemple de caractériser certaines classes de treillis, ou encore d'en concevoir des représentations compactes du treillis [66, 67, 72, 78]. Un ouvrage récent [49] présente les fondements des structures ordonnées.

1990 : Treillis des concepts et Analyse Formelle des Concepts Le *treillis des concepts* a été introduit dans les années 1980 par Wille dans le cadre de l'Analyse Formelle des Concepts (AFC)[82], avec un ouvrage de référence datant de 1999 [58]. L'AFC est une approche à la représentation des connaissances en pleine émergence dans les années 90 qui définit le treillis des concepts à partir de données binaires (ou *contexte*) de types objet $\times$ attributs. Un noeud du treillis, appelé *concept*, est un regroupement maximal d'objets possédant des attributs en commun. Le treillis ainsi composé de concepts reliés par inclusion, fournit une représentation des données très intuitive. L'émergence du treillis des concepts ces dernières années s'explique à la fois par la part grandissante de l'informatique dans la plupart des champs disciplinaires, ce qui conduit à la production de données en quantité de plus en plus importante ; mais également par la récente montée en puissance des ordinateurs qui, bien que la taille du treillis puisse être exponentielle en fonction des données dans le pire des cas, rend possible

le développement d'un grand nombre d'applications le concernant. Des extensions du champ de l'AFC à des données plus complexes que les données binaires ont récemment été proposées [75][55][57], extensions qui maintiennent une connexion de Galois entre les objets et leur espace de description.

2.2 Treillis ordinal et treillis algébrique

2.2.1 Définitions algébrique et ordinale d'un treillis

On trouve dans la littérature deux définitions d'un treillis : une définition algébrique et une définition ordinale. Ces définitions introduisent toutes deux la notion de *borne supérieure* (ou *supremum*) et de *borne inférieure* (ou *infimum*) : alors qu'il s'agit d'opérateurs binaires dans la définition algébrique, ce sont des éléments particuliers dans la définition ordinale. La figure 2.1(b) en fournit un exemple.

Définition 1 (Définition algébrique) *[47] Un* treillis *ou encore une* algèbre de treillis *est un triplet $L = (S, \vee, \wedge)$ où $\vee$ et $\wedge$ sont deux opérateurs binaires sur l'ensemble S qui vérifient les propriétés suivantes :*

- *associativité : pour tous $x, y, z \in S$, $(x \vee y) \vee z = x \vee (y \vee z)$ et $(x \wedge y) \wedge z = x \wedge (y \wedge z)$*
- *commutativité : pour tous $x, y \in S$, $x \vee y = y \vee x$ et $x \wedge y = y \wedge x$*
- *idempotence : pour tous $x \in S$, $x \vee x = x = x \wedge x$*
- *loi d'absorption : pour tous $x, y \in S$, $x \vee (x \wedge y) = x = x \wedge (x \vee z)$*

Définition 2 (Définition ordinale) *[45] Un treillis est une paire $L = (S, \leq)$ où :*

- *$\leq$ est une relation d'ordre sur l'ensemble S, i.e. une relation binaire qui vérifie les propriétés suivantes :*
  - *réflexivité : pour tout $x \in S$, on a xRx*
  - *antisymétrie : pour tous $x, y \in S$, xRy et yRx impliquent $x = y$*
  - *transitivité : pour tous $x, y, z \in S$, xRy et yRz impliquent xRz*
- *toute paire d'éléments $\{x, y\}$ de S admet à la fois une borne inférieure et une borne supérieure :*
  - *la* borne inférieure *de x et y, notée $x \wedge y$, est l'unique* élément maximal *(plus grand élément) de l'ensemble des* prédécesseurs *(ou* minorants*) de x et y (ensemble des éléments $z \in S$ tels que $z \leq x$ et $z \leq y$).*
  - *la* borne supérieure *de x et y, notée $x \vee y$, est l'unique* élément minimal *(i.e. plus petit élément) de l'ensemble des* successeurs *(ou* majorants*) de x et y (ensemble des éléments $z \in S$ tels que $z \geq x$ et $z \geq y$).*

Les bornes inférieure et supérieure se définissent de façon plus générale mais équivalente pour une partie $X \subseteq S$: la borne inférieure de X, notée $\wedge X$, est l'unique élément maximal de l'ensemble des minorants de X, alors que la borne supérieure de X, notée $\vee X$, est l'unique élément minimal de l'ensemble des majorants de X.

Lorsque seule l'existence de la borne inférieure est vérifiée, on parle d'*inf-demi-treillis*. Un *sup-demi-treillis* est défini dans le cas dual où seule l'existence d'une borne supérieure est vérifiée. Un treillis est donc à la fois un inf-demi-treillis et un sup-demi-treillis. L'ensemble des noeuds d'un arbre muni de la relation "est ancêtre de" forme ainsi un sup-demi-treillis où aucune borne inférieure d'éléments incomparables n'existe, alors qu'il forme un inf-demi-treillis lorsqu'on considère la relation "est descendant de".

On montre facilement qu'un treillis admet un unique élément maximal noté $\top$, ainsi qu'un unique élément minimal noté $\bot$. On montre également qu'un inf-demi-treillis muni d'un unique élément maximal est un treillis, de même que l'est un sup-demi-treillis muni d'un unique élément minimal.

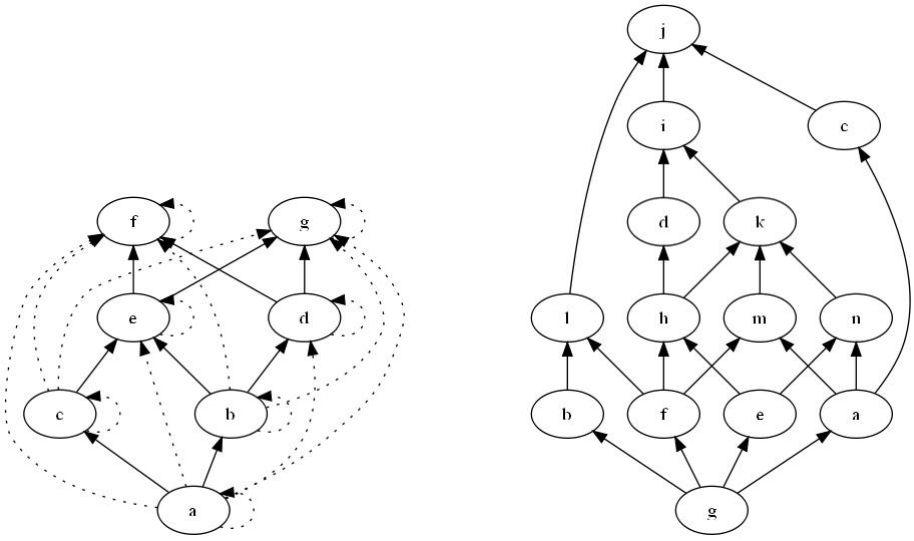
A toute relation d'ordre $\leq$ on associe sa *relation d'ordre strict* : notée $<$, c'est une relation asymétrique ($(x, y) \in R$ implique $(y, x) \notin R$) transitive et irréflexive définie par $x < y$ si $x \leq y$ et $x \neq y$. Elle se déduit de la relation $\leq$ en supprimant les relations de réflexivité.

On associe également à une relation d'ordre sa *relation de couverture* : notée $\prec$, c'est une relation antisymétrique définie par $x \prec y$ si $x < y$, et il n'existe pas d'élément z tel que $x < z < y$. On dit alors que y *couvre* x. Elle se déduit de la relation $\leq$ en supprimant les relations de réflexivité et de transitivité.

Il est d'usage d'identifier relation binaire et graphe orienté simple (i.e. un graphe sans arcs multiples) où chaque élément est représenté par un sommet du graphe, et où la relation entre deux éléments x et y est représentée par un arc du graphe entre le sommet x et le sommet y. Le *diagramme de Hasse* est une représentation d'un ordre proche de la représentation habituelle d'un graphe : les noeuds sont positionnés de telle sorte que x sera au-dessous de y lorsque $x \leq y$; les orientations des arcs ne sont pas toujours représentées car elles peuvent se déduire du positionnement des noeuds ; seuls les arcs de la relation de couverture $\prec$ sont représentés car les arcs de réflexivité (boucles) et de transitivité s'en déduisent. Le diagramme de Hasse permet ainsi de ne pas surcharger le dessin pour faciliter une meilleure lisibilité. Il s'étend naturellement au cas particulier des treillis.

Exemple 1 *Un ordre est représenté Figure 2.1(a) avec les relations de transitivité et de réflexivité en pointillés, relations qui ne sont pas conservées dans son diagramme de Hasse pour une meilleure lisibilité. Bien que les éléments b et c possèdent e pour borne supérieure et a pour borne inférieure, les éléments e et d n'admettent pas de borne supérieure, et les éléments f et g pas de borne inférieure. Par conséquent, cet ordre n'est pas un treillis. Il possède a pour unique élément minimal, et f et g pour éléments maximaux.*

Un exemple de treillis est fourni Figure 2.1(b). On peut vérifier que toute paire d'éléments admet une borne supérieure ainsi qu'une borne inférieure. De plus, ce treillis posséde g pour unique élément minimal, également appelé $\bot$, et j pour unique élément maximal, également appelé $\top$.



(a) Exemple d'ordre (b) Exemple de treillis

FIGURE 2.1 – Exemples d'ordre et de treillis

Un treillis est *distributif* si $\vee$ et $\wedge$ vérifient la propriété de distributivité : pour tous $x, y, z \in S$, $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$. Un treillis est *complémenté* si tout élément $x \in S$ admet au moins un *complément*, i.e. un élément $x' \in S$ tel que $x \vee x' = \top$ et $x \wedge x' = \bot$. Si, pour tout élément $x \in S$, il existe un $\vee$-complément (i.e. élément $x' \in S$ tel que $x \vee x' = \top$), le treillis est $\vee$-complémenté. La définition d'un treillis $\wedge$-complémenté est duale. Un treillis est *booléen* s'il est à la fois distributif et complémenté.

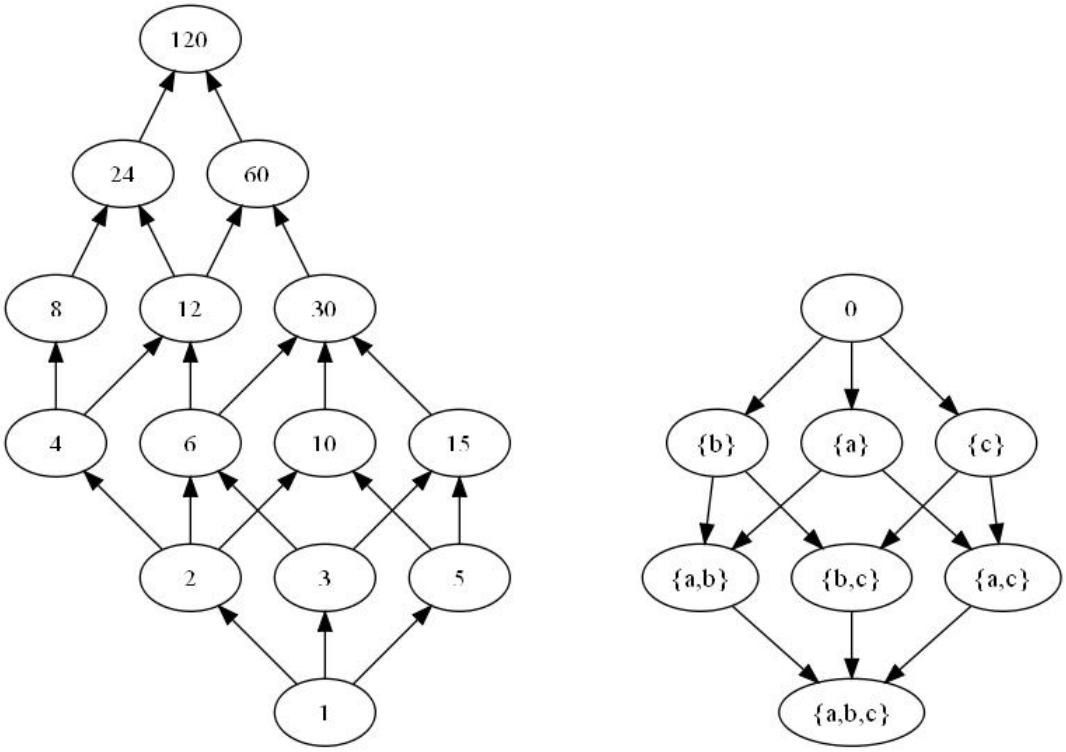
Un exemple classique de treillis distributif est un ensemble fini de nombres muni des deux opérateurs $\wedge = pgcd$ et $\vee = ppcm$ (cf Figure 2.2(a)). La relation d'ordre est alors la relation "divise". L'ensemble $\mathcal{P}(\mathcal{S})$ des parties d'un ensemble S muni de la relation d'inclusion est quant à lui un exemple classique de treillis booléen, i.e. de treillis distributif et complémenté ; la relation d'inclusion est un ordre, $\vee = \cup$ et $\wedge = \cap$ (cf Figure 2.2(b)).

Dans la suite de ce document, pour une meilleure lisibilité, les ensembles seront assimilés à des mots. L'ensemble $\{a, b\}$ sera ainsi noté ab. Par abus de notation, l'union ensembliste $X \cup \{x\}$ sera notée $X + x$, et la différence ensembliste $X \backslash \{x\}$ sera notée $X - x$.

2.2.2 Irréductibles et générateurs minimaux d'un treillis

Un élément d'un treillis est dit *réductible* s'il est résultat de l'application de l'opérateur de borne supérieure $\vee$ ou inférieure $\wedge$ sur au moins deux autres éléments distincts. Dans le cas contraire, il sera dit *irréductible*.

Définition 3 (Elements irréductibles) *Soit $L = (S, \leq)$ un treillis.*



(a) Exemple de treillis distributif (b) Exemple de treillis booléen

FIGURE 2.2 – Exemples de treillis

- *Un élément $j \in S$ est appelé* sup-irréductible *si, pour tout sous-ensemble X de S, $j = \vee X$ implique que $j \in X$.*
- *Un élément $m \in S$ est appelé* inf-irréductible *si, pour tout sous-ensemble X de S, $j = \wedge X$ implique que $j \in X$.*

L'ensemble des sup-irréductibles du treillis L est noté J_L, et celui des inf-irréductibles M_L.

En particulier, on a $\bot = \vee\emptyset$ et $\top = \wedge\emptyset$, impliquant ainsi que $\bot$ n'est pas un sup-irréductible, et $\top$ n'est pas un inf-irréductible.

Tout élément réductible peut par conséquent s'obtenir par applications successives des opérateurs $\vee$ et $\wedge$ sur un ensemble d'irréductibles. Les éléments irréductibles représentent alors la structure même du treillis, la plupart des résultats structurels pouvant s'établir sur la base de leurs propriétés, notamment les isomorphismes entre treillis, treillis de fermés et treillis de Galois.

Une caractérisation immédiate des éléments irréductibles à partir de la relation de couverture $\prec$ du treillis, et donc de son diagramme de Hasse, établit qu'un élément est un sup-irréductible si et seulement si il couvre un seul élément, noté j^-, alors qu'un élément est un inf-irréductible si et seulement si il est couvert pas un seul élément, noté m^+. En particulier, les *atomes* (un atome est un élément couvrant l'élément minimal $\bot$)

sont des sup-irréductibles, alors que les *co-atomes* (un co-atome est un élément couvert pas l'élément maximal $\top$) sont des inf-irréductibles. Lorsque tous les sup-irréductibles sont des atomes, le treillis est alors un *treillis atomistique*. Dualement, lorsque tous les inf-irréductibles sont des co-atomes, le treillis est un *treillis co-atomistique*.

Tout élément $x \in S$ d'un treillis $L = (S, \leq)$ est la borne supérieure de l'ensemble de ses prédécesseurs, ainsi que la borne inférieure de l'ensemble de ses successeurs. La définition latticielle permet d'établir facilement qu'il est possible de réduire ces ensembles aux seuls prédécesseurs sup-irréductibles, et successeurs inf-irréductibles :

$$x = \vee J_x = \vee\{y \text{ sup-irréductible tel que } y \leq x\} \quad (2.1)$$

$$x = \wedge M_x = \vee\{y \text{ inf-irréductible tel que } y \geq x\} \quad (2.2)$$

Par conséquent, les ensembles d'éléments irréductibles portent l'information minimale permettant de reconstruire le treillis dans sa globalité par reconstructions successives de bornes supérieures (en utilisant les sup-irréductibles) ou de bornes inférieures (en utilisant les inf-irréductibles).

Cependant, un ensemble minimal nécessaire à la reconstruction par borne supérieure d'un élément donné $x \in S$ peut ne pas être l'ensemble complet J_x, mais en être une ou plusieurs parties strictes. On parle alors de générateurs minimaux de x :

Définition 4 *Soit $L = (S, \leq)$ un treillis et $x \in S$ un élément du treillis. Un* générateur minimal *de x est un sous-ensemble B de J_x tel que $x = \vee B$ et qui soit minimal au sens de l'inclusion, i.e pour tout $A \subset B$, on a $x < \vee A$. La famille $\mathcal{B}_x$ des générateurs minimaux de x se définit donc par :*

$$\mathcal{B}_x = \{B \subseteq J_x \ : \ x = \vee B \text{ et } x < \vee A \text{ pour tout } A \subset B\} \quad (2.3)$$

L'observation duale vaut pour la reconstruction de x comme borne inférieure à partir de M_x. Le nombre de générateurs minimaux d'un élément x peut se réduire à 1 (le seul ensemble J_x), mais peut également être élevé, au pire exponentiel en la cardinalité de J_x, d'où une explosion combinatoire de la cardinalité de $\mathcal{B}_x$.

Exemple 2 *Considérons l'exemple du treillis de la figure 2.1(b). Six éléments possèdent un seul arc entrant, et forment l'ensemble des sup-irréductibles alors que les inf-irréductibles, caractérisés, quant à eux, par un seul arc sortant, sont au nombre de sept :*

$$J = \{a, b, c, d, e, f\} \quad (2.4)$$

$$M = \{b, c, d, i, k, l, m, n\} \quad (2.5)$$

Quatre sup-irréductibles sont également des atomes $\{a, b, e, f\}$, alors que trois inf-irréductibles sont également des co-atomes $\{c, i, l\}$. Ces éléments irréductibles sont utilisés dans le treillis de la Figure 2.3 où, dans le noeud de chaque élément x sont précisés

x	a	b	c	d	e	f	g
J_x	a	b	ac	def	e	f	$\emptyset$
$\mathcal{B}_x$	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$	$\{f\}$	$\{\emptyset\}$
x	h	i	j	k	l	m	n
J_x	ef	adef	abcdef	aef	bf	af	ae
$\mathcal{B}_x$	$\{ef\}$	$\{ad\}$	$\{ab, bc, bd, dc, be\}$	$\{aef\}$	$\{bf\}$	$\{af\}$	$\{ae\}$

TABLE 2.1 – Générateurs minimaux des éléments du treillis de la Figure 2.1(b)

l'ensemble J_x des sup-irréductibles inférieurs ou égaux à x ainsi que l'ensemble M_x des inf-irréductibles supérieurs ou égaux à x.

Les générateurs minimaux de chaque élément sont, quant à eux, donnés par la table 2.1. On peut ainsi observer que chaque sup-irréductible est son propre générateur minimal, mais aussi que tous les éléments, excepté l'élément maximal $\top$, ont pour unique générateur minimal l'ensemble de leurs prédécesseurs sup-irréductibles. L'élément maximal possède quant à lui 5 générateurs minimaux.

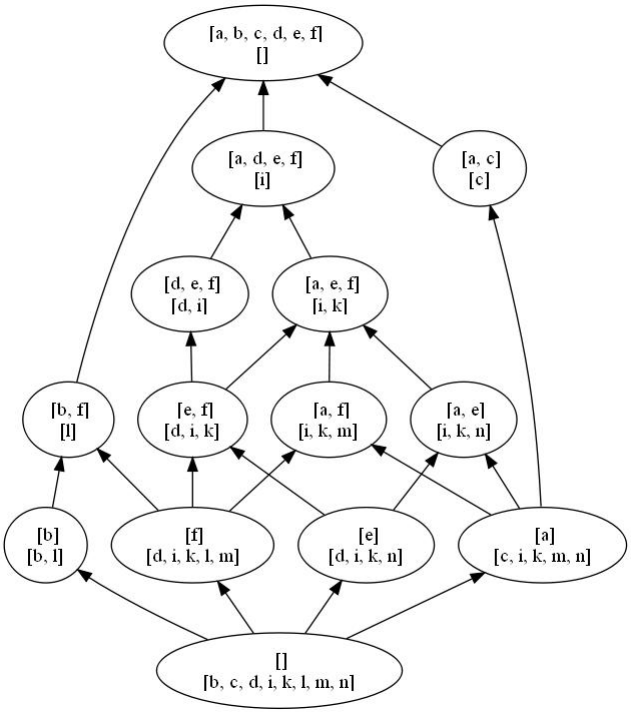


FIGURE 2.3 – Treillis de la figure 2.1(b) où sont précisés, pour chaque élément x, les ensembles J_x et M_x

Une *complétion* d'un ordre quelconque en un treillis consiste à ajouter dans l'ordre des bornes supérieures et inférieures lorsqu'elles n'existent pas. Une complétion est mi-

nimale lorsqu'un nombre minimal d'éléments sont ajoutés. Connue sous la terminologie de complétion de Dedekind-MacNeille, la complétion minimale d'un ordre en un treillis est connue pour être unique [102], et se caractérise à l'aide des éléments irréductibles. Il existe plusieurs algorithmes efficaces pour la calculer [86, 104]. Toute la difficulté consiste à ne pas ajouter d'éléments superflus, difficulté illustrée par la figure 2.4 : après ajout des deux bornes manquantes $e \wedge f$ puis $f \wedge g$, la complétion obtenue n'est pas minimale.

En collaboration avec Lhouari Nourine, nous avons proposé un algorithme de génération à la demande de la complétion minimale [15] qui repose sur un codage des éléments de l'ordre initial sur la base des irréductibles, codage qui préserve les bornes inf et sup.

Nous avons également proposé une caractérisation du plus petit sous-treillis d'un treillis contenant certains de ses éléments [12]. Là encore, cette caractérisation repose sur les propriétés des éléments irréductibles et induit un algorithme de calcul. On peut également citer quelques propositions algorithmiques où les propriétés ordinales du treillis sont utilisées pour calculer les extensions faibles d'un ordre [13, 14], ou encore le pléthysme de certaines fonctions quasi-symétriques [10].

2.2.3 Table et graphe de dépendance d'un treillis

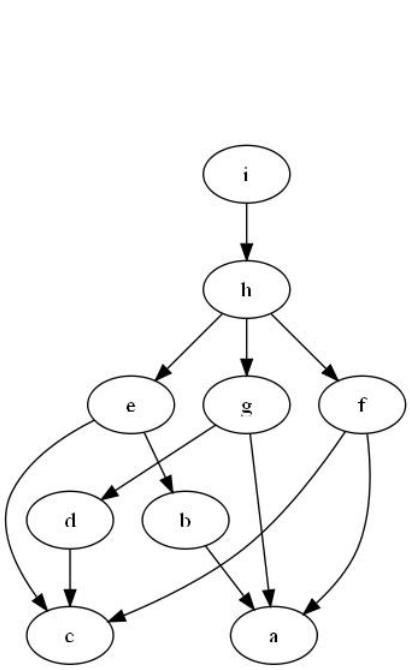
La notion d'éléments irréductibles d'un treillis permet de concevoir des représentations compactes du treillis qui contiennent l'information nécessaire à sa reconstruction : représentation par le sous-graphe des irréductibles ; représentation par un ordre biparti introduit par Markowski [66, 67] où les sup-irréductibles sont les éléments maximaux, et les inf-irréductibles sontles éléments minimaux ; représentation par une table composée en colonne des sup-irréductibles et en ligne des inf-irréductibles. On distinguera la table binaire de la table complète contenant les relations flèches.

Plus précisemment, la table est

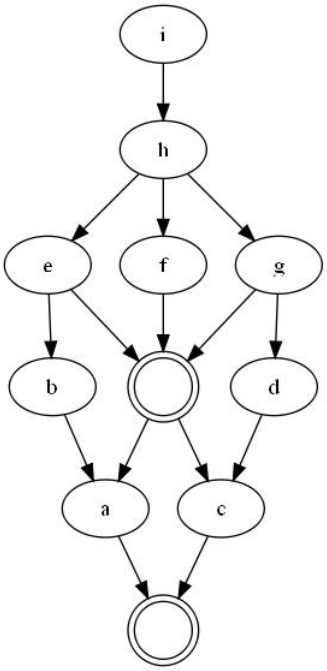
La reconstruction par concepts à partir de la table binaire (cf Section 3.1.1) est la plus commune. On peut citer également la reconstruction par antichaînes maximales ou idéaux maximaux à partir du *biparti des irréductibles* [106, 104].

On peut cependant trouver d'autres mécanismes de reconstruction par fermeture (cf section 2.3) à partir des seuls sup-irréductibles organisés sous forme d'un graphe valué - le *graphe de dépendance d'un treillis* - sous forme de règles d'implication (cf section 3.2) ou encore sous forme d'une fonction booléenne [89], structures dont la taille peut être exponentielle en celle de l'ensemble des sup-irréductibles.

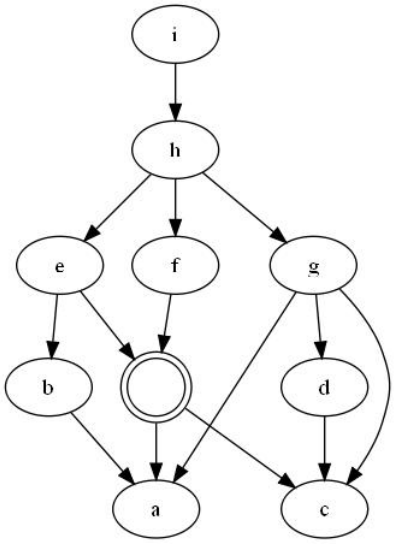
La *table* d'un treillis se définit à partir des *relations flèches* qui partitionnent les différents liens possibles entre un sup-irréductible et un inf-irréductible à l'aide de cinq relations binaires définies sur $J_L \times M_L$. La relation de comparabilité $\leq$ permet tout



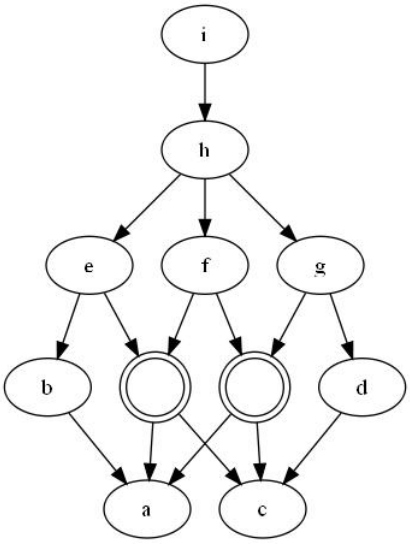
(a) Ordre inital



(b) Complété de Dedekind-Macneille



(c) Ajout de $e \wedge f$



(d) Ajout de $f \wedge g$

FIGURE 2.4 – Création d'éléments superflus

d'abord d'établir une première partition de $J_L \times M_L$ en deux parties notées $P_{\leq}$ et $P_{\not\leq}$:

$$P_{\leq} \quad = \quad \{(j, m) \in J_L \times M_L \ : \ j \leq m\} \tag{2.6}$$

$$P_{\not\leq} \;=\; \{(j,m) \in J_L \times M_L \;:\; j \not\leq m\} \tag{2.7}$$

Les deux *relations flèches* $\uparrow$ et $\downarrow$ permettent d'affiner les liens entre les paires $(j,m) \in P_{\not\leq}$ d'irréductibles incomparables.

Définition 5 (Relations flèches) *[83] Soit $L = (S, \leq)$ un treillis, $j \in L$ et $m \in M_L$.*
- *$j \uparrow m$ si $j \not\leq m$ et $j < m^+$.*
- *$j \downarrow m$ si $j \not\leq m$ et $j^- < m$.*

Ces deux relations n'étant pas disjointes, considérer les quatre combinaisons possibles permet de partitionner l'ensemble des paires $(j,m) \in P_{\not\leq}$ en quatre parties $P_\uparrow$; $P_\downarrow$, $P_\updownarrow$ et $P_\circ$:

$$P_\uparrow \;=\; \{(j,m) \in J_L \times M_L \;:\; j \uparrow m \text{ et } j \not\downarrow m\} \tag{2.8}$$
$$P_\downarrow \;=\; \{(j,m) \in J_L \times M_L \;:\; j \not\uparrow m \text{ et } j \downarrow m\} \tag{2.9}$$
$$P_\updownarrow \;=\; \{(j,m) \in J_L \times M_L \;:\; j \uparrow m \text{ et } j \downarrow m\} \tag{2.10}$$
$$P_\circ \;=\; \{(j,m) \in J_L \times M_L \;:\; j \not\uparrow m \text{ et } j \not\downarrow m\} \tag{2.11}$$

La *table* d'un treillis se définit comme une représentation tabulaire de ce partitionnement :

Définition 6 (Table d'un treillis) *La* table *T d'un treillis $L = (S, \leq)$ est composée d'une colonne par sup-irréductible, d'une ligne par inf-irréductible. Pour $m \in M_L$ et $j \in J_L$, $T[m,j]$ contient $\times, \uparrow, \downarrow, \updownarrow$ ou $\circ$ selon que (m,j) appartienne à $P_\leq$, $P_\uparrow$, $P_\downarrow$, $P_\updownarrow$ ou $P_\circ$.*

Le partitionnement en deux parties $P_\leq$ et $P_{\not\leq}$ induit quant à lui une représentation d'un treillis par sa *table binaire* où $T[m,j]$ contient $\times$ si (m,j) appartient à $P_\leq$, et rien sinon (i.e. (m,j) appartient à $P_{\not\leq}$).

La table binaire du treillis décrit suffisamment les liens entre irréductibles pour permettre sa reconstruction. Un résultat important établit que tout treillis est isomorphe au *treillis de Galois*, ou *treillis des concepts* de sa table binaire. Dans le cadre de l'analyse formelle des concepts, la table binaire se retrouve sous la dénomination de *contexte réduit*, à partir duquel est défini son *treillis des concepts* (voir section 3.1.2).

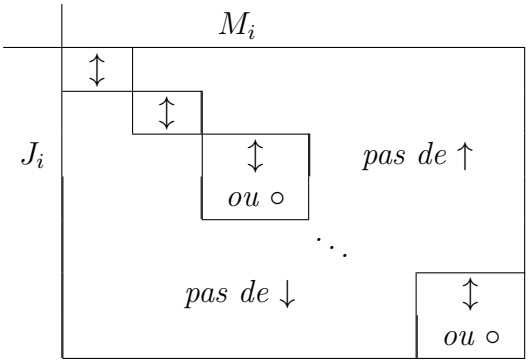
La table complète du treillis contient quant à elle une description plus fine des liens entre irréductibles, utilisée essentiellement pour caractériser certaines propriétés des treillis. Ainsi, il apparaît que la table d'un treillis contient toujours au moins une double flèche $\updownarrow$ sur chaque ligne et sur chaque colonne. Lorsque la table possède exactement une double flèche sur chacune de ses lignes et colonnes, alors le treillis est distributif. Un treillis atomistique se caractérise quant à lui par l'absence de flèche $\uparrow$ dans sa table, alors qu'un treillis co-atomistique se caractérise par l'absence de flèche $\downarrow$. Par conséquent, la

table d'un treillis booléen, qui rappelons-le est à la fois distributif, atomistique et co-atomistique, se caractérise par une unique double flèche $\updownarrow$ sur chacune de ses lignes et colonne, ainsi que par l'absence de flèches simples $\uparrow$ et $\downarrow$.

La classe $\mathcal{CN}$ est la classe des treillis dupliqués par convexes, classe de treillis introduite dans [53] et composée de tous les treillis que l'on peut obtenir incrémentalement, à partir du treillis à deux éléments $\mathcal{Q} = (\{0,1\}, \leq)$ par duplication d'une partie connexe et convexe. Plus formellement, il s'agit de la classe des treillis T tel qu'il existe une séquence $\mathcal{Q} = T_1, ..., T_i, ..., T_{p-1}, T_p = T$ de treillis ainsi qu'une séquence $C_1, ..., C_i, ..., C_{p-1}$ telle que C_i est un ensemble convexe de T_i et T_{i+1} est obtenu par duplication à partir de T_i, pour tout $i < p$, selon la règle de duplication introduite par Day dans [52]. Plus précisément, cette règle de duplication définit le treillis $T' = T[C]$ comme résultat de la duplication dans T d'une convexe C de T par l'ensemble $X' = (X - C) \cup (C \times \mathcal{Q})$ et la relation $\leq_{T'}$ avec, pour x' et $y' \in X'$, par :

$$x' \leq_{T'} y' \iff \begin{cases} x', y' \in X - C \text{ et } x' \leq_T y' \text{ ou} \\ x' \in X - C,\ y' = yi \in C \times \mathcal{Q} \text{ et } x' \leq_T y, \text{ ou} \\ x' = xi \in C \times \mathcal{Q},\ y' \in X - C \text{ et } x \leq_T y', \text{ ou} \\ x' = xi \in C \times \mathcal{Q},\ y' = yj \in C \times \mathcal{Q},\ x \leq_T y \text{ et } i \leq j \end{cases}$$

En collaboration avec Nathalie Caspard [6], nous avons proposé un théorème de caractérisation des treillis de la classe $\mathcal{CN}$: un treillis est dans la classe $\mathcal{CN}$ si et seulement si sa table peut s'organiser sous la forme suivante qui décrit une séquence de duplications de sous-ensembles convexes et connexes $C_T = (C_i)_{1<i\leq p}$ permettant d'obtenir T depuis $\mathcal{Q}$:



c'est-à-dire s'il satisfait les conditions suivantes :

$$\begin{cases} (a) & \text{il existe un escalier de marches généré par } C_T \\ & \text{et chaque marche } (J_i \times M_i) \text{ ne contient que des } \updownarrow \text{ ou des } \circ, \\ (b) & \text{toutes les } \downarrow \text{ sont au dessus de cet escalier,} \\ (c) & \text{toutes les } \uparrow \text{ sont en dessous de cet escalier.} \end{cases}$$

De cette caractérisation d'un treillis $\mathcal{CN}$ par sa table nous avons déduit un algorithme de reconnaissance polynomial des treillis de la classe $\mathcal{CN}$. Cette opération de duplication est une généralisation de la duplication d'intervalles qui caractérise les treillis

bornés [52], mais aussi les treillis booléens - duplication de l'intervalle contenant le treillis complet - ainsi que les treillis distributifs - duplication de l'intervalle contenant l'élément maximal ou minimal.

Une autre structure moins connue est celle du graphe de dépendance *d'un treillis, que j'ai introduit dans mes travaus de thèse [2], et à partir duquel sa reconstruction est possible. Il s'agit d'un graphe valué dont les arcs, qui correspondent à la relation de dépendance entre sup-irréductibles introduite dans [68], sont valués par les générateurs minimaux. Le nombre de générateurs minimaux étant exponentiel dans le pire des cas, celle du graphe de dépendance peut l'être également.*

Définition 7 (Graphe de dépendance) *Soit $L = (S, \leq)$ un treillis. Le graphe de dépendance du treillis L est un graphe valué $G = (J_L, \delta, \omega)$ avec :*

- *δ* relation de dépendance *[68, 70] définie sur J_L par $j\delta j'$ si il existe $x \in S$ tel que $j \not\leq x$, $j' \not\leq x$ et $j < j' \vee x$. On notera $j\delta_x j'$.*
- *ω valuation des arcs définie pour chaque relation $j\delta j'$ par*

$$\omega(j, j') = \{\textit{générateurs minimaux de } x \ : \ j\delta_x j' \textit{ et } x \textit{ minimal dans le treillis}\}$$

On pourra noter $(j, j') \in \delta$ par $j\delta_x j'$ ou encore par $j\delta_B j'$, avec B générateur minimal de x. Ainsi, $j\delta j'$ s'il existe $x \in S \setminus \{j, j'\}$ tel que $j\delta_x j'$. Ou encore $\delta = \cup\{\delta_x \ : \ x \in S\}$.

La restriction du graphe de dépendance d'un treillis à la seule relation $\delta_\emptyset$ correspond au sous-graphe du treillis induit par ses sup-irréductibles car $j < j'$ implique $j < j' \vee \perp$ et $\omega(j, j') = J_\perp = \{\emptyset\}$. De plus, on montre facilement qu'un treillis est distributif si et seulement si les valuations de son graphe de dépendance ne contiennent que l'ensemble vide.

J'ai proposé d'extraire du graphe de dépendance un ensemble de règles d'implication *- une règle d'implication est composée de deux sous-ensembles d'éléments appelés prémisse et conclusion - permettant la reconstruction du treillis : à toute relation $j\delta_X j'$ on associe la règle d'implication $X + j' \to j$. Le mécanisme de reconstruction du treillis est une conséquence d'un résultat structurel établissant que tout treillis est isomorphe au treillis de fermés du système de règles porté par son graphe de dépendance (voir section 3.2).*

Un autre résultat (voir section 3.2.2) établit que la prémisse $X + j'$ de chaque règle d'implication $X + j' \to j$ est un générateur minimal de $\vee X + j'$, qui se définit ainsi récursivement à partir de X, générateur minimal de x. Nous avons exploité cette définition récursive des générateurs minimaux d'un point de vue algorithmique.

L'inconvénient majeur de cette représentation de graphe de dépendance réside cependant dans sa taille, ou plus précisément dans celle de ses arcs, car le nombre de générateurs minimaux peut être exponentiel en celle des sup-irréductibles du treillis. Le nombre du noeuds du graphe reste quant à lui polynomial, mais n'induit pas de mécanisme de reconstruction.

	a	b	c	d	e	f
b	↓	×	○	○	↓	↕
c	×	↕	×	↑	↕	↕
d	↕	↓	○	×	×	×
i	×	↕	↕	×	×	×
k	×	↓	↓	↕	×	×
l	↕	×	↑	↑	↕	×
m	×	↓	↓	○	↕	×
n	×	↓	↓	○	×	↕

TABLE 2.2 – Table du treillis de la figure 2.1(b)

Exemple 3 *Les éléments irréductibles du treillis de la figure 2.1(b)sont utilisés pour en décrire la structure à l'aide de sa table (cf table 2.2), ou encore de son graphe de dépendance (cf Figure 3).*

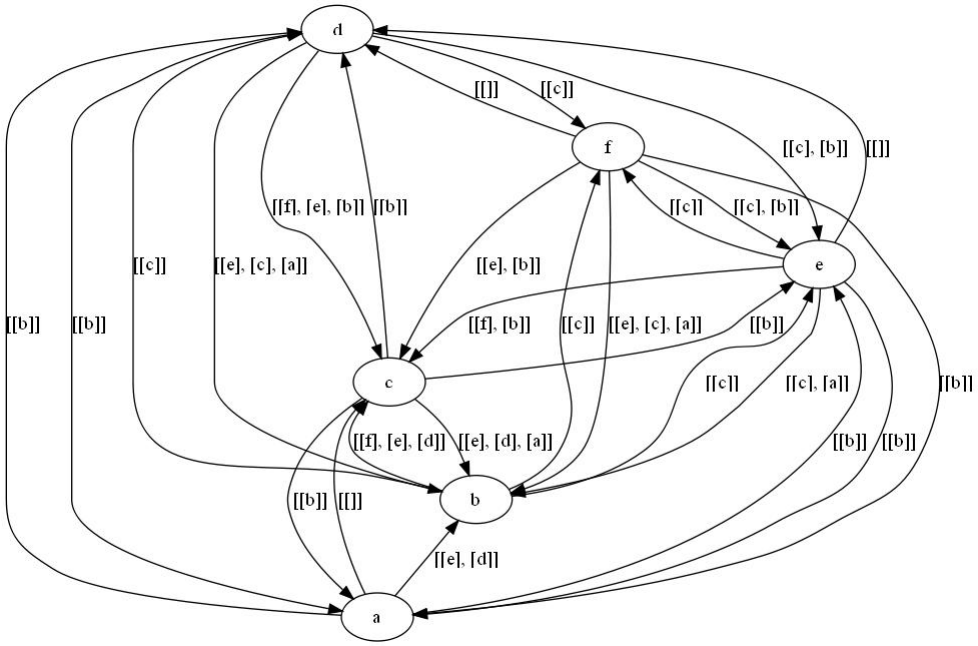


FIGURE 2.5 – Graphe de dépendance du treillis de la figure 2.1(b) défini sur l'ensemble de ses sup-irréductibles

La représentation d'un treillis sous la forme d'un graphe valué défini sur les seuls sup-irréductibles a été initialement introduite dans [72] sous la terminologie de OD-Graph. Il s'agit d'un graphe défini de façon similaire au graphe de dépendance, à partir de la relation de dépendance forte Δ d'un treillis, et dont les arcs sont également valués par

des parties de sup-irréductibles appelées les *couvertures minimales*. Plus précisément, la relation de dépendance forte, inclue dans la relation de dépendance, se définit sur J_L par $j\Delta j'$ si il existe $x \in S$ tel que $j < j' \vee x$ et $j \not< j'^{-} \vee x$.

Le OD-Graph a tout d'abord été exploité dans l'étude des treillis bornés inférieurs [56]. Il sert de base à des travaux plus récents sur la théorie de la dualité [78]. Il définit également un système de règles à partir duquel la reconstruction du treillis est possible, selon un mécanisme plus complexe que la reconstruction à partir du graphe de dépendance. Cependant, à la différence du graphe de dépendance, la taille des valuations qui le définissent reste polynomiale en celle du treillis.

2.3 Treillis de fermés et système de fermeture

Une *famille de Moore* est une famille $\mathcal{F}$ de parties - dites *fermés* - d'un ensemble S stable par intersection et contenant S lui-même. $\mathcal{F}$ munie de la relation d'inclusion est un treillis. On parle alors de *treillis de fermés* pour le treillis. Comme tout treillis est représentable par un treillis de fermés, tout treillis peut se manipuler de façon équivalente sous forme algébrique, ordinale, ou encore ensembliste. L'apport principal du treillis de fermés réside dans le lien qui l'unit à un *système de fermeture* via un *opérateur de fermeture*.

2.3.1 Treillis de fermés

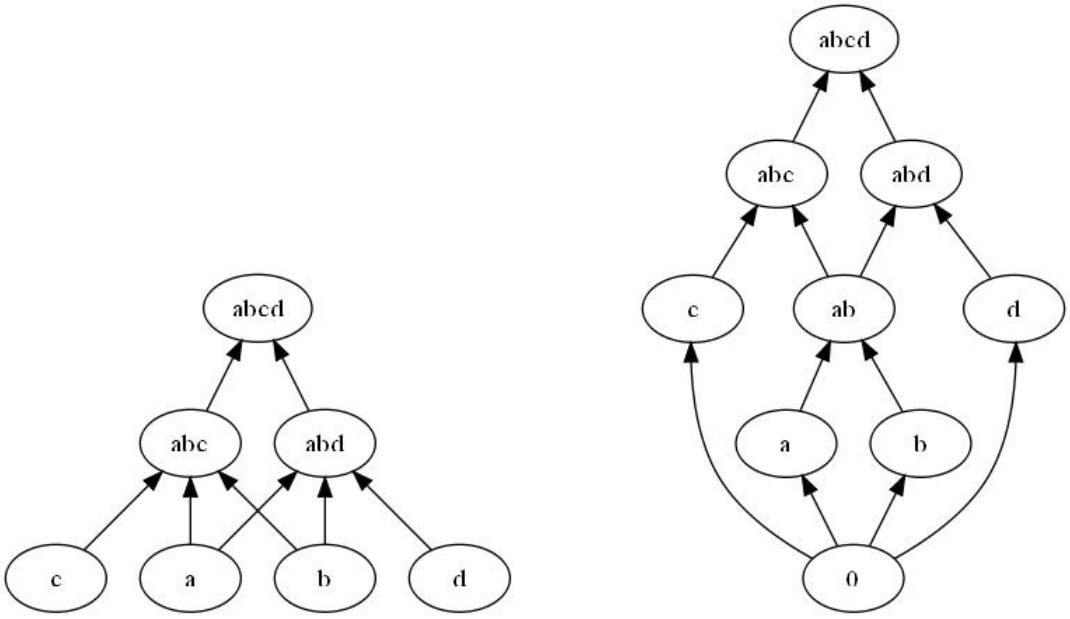
Toute famille $\mathcal{F} \in \mathcal{P}(S)$ munie de la relation d'inclusion est ordonnée, l'inclusion étant transitive, réflexive et antisymétrique (cf Figure 2.6). La propriété de treillis est quant à elle garantie pour une famille possédant les propriétés d'une famille de Moore [71]. On parle alors de treillis de fermés.

Définition 8 (Treillis de fermés) *Un treillis de fermés sur un ensemble S est une paire $(\mathcal{F}, \subseteq)$ où $\mathcal{F}$ est une famille sur S possèdant les propriétés d'une* famille de Moore*, encore appelée* famille de fermés *:*

- *$\mathcal{F}$ contient S*
- *$\mathcal{F}$ est stable par intersection : pour tous $F, F' \in \mathcal{F}$, on a $F \cap F' \in \mathcal{F}$*

En particulier, $\mathcal{F} = \mathcal{P}(S)$ est une famille de Moore, et la structure de treillis induite $(\mathcal{P}(S), \subseteq)$ est un treillis booléen (cf Figure 2.2(b)). Un autre exemple de famille de Moore est celui obtenu à partir des hiérarchies. Une hiérarchie est une famille $\mathcal{F}$ sur S contenant S lui-même, ainsi que tous les singletons, et telle que, pour F et F' parties de la famille, la propriété suivante soit vérifiée :

$$F \cap F' = \emptyset \text{ ou } F \subseteq F' \text{ ou } F' \subseteq F \tag{2.12}$$



(a) Relation d'ordre (b) Treillis

FIGURE 2.6 – Deux familles ordonnées par inclusion

En ajoutant l'ensemble $\emptyset$ à cette hiérarchie, on obtient une famille de Moore. La structure de treillis induite est un arbre auquel un élément minimal relié à ses feuilles a été ajouté.

Bien que les propriétés d'une famille de Moore sur un ensemble S sont purement ensemblistes (stabilité par intersection et appartenance de l'ensemble S lui-même), ses propriétés latticielles permettent d'introduire les opérateurs de bornes inférieures $\wedge$ et supérieures $\vee$, pour F et F' parties de $\mathcal{F}$:

$$F \wedge F' = F \cap F' \tag{2.13}$$

$$F \vee F' = \cap\{F'' \in \mathcal{F} \ : \ F \cup F' \subseteq F''\} \tag{2.14}$$

Un résultat fondamental de la théorie des treillis permet de manipuler de façon équivalente treillis et treillis de fermés dans le cas fini :

Bijection 1 *Treillis et treillis de fermés sont en correspondance bijective.*

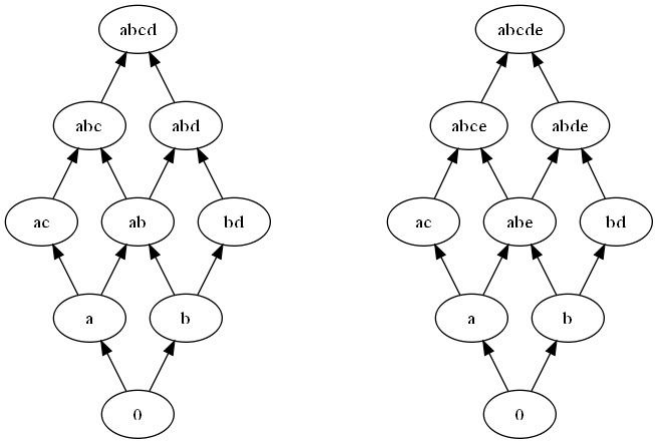
Cette bijection s'exprime sur la base des éléments sup-irréductibles : à tout treillis $(S, \leq)$ est associé le treillis de fermés $((J_x)_{x\in S}, \subseteq)$ composé de la famille de Moore $(J_x)_{x\in S}$ définie sur l'ensemble des sup-irréductibles du treillis. Il a été établi qu'il n'existe pas d'ensemble de cardinalité inférieure à celle de l'ensemble des sup-irréductibles à partir duquel définir un treillis de fermés isomorphe à $(S, \leq)$. La famille $(J_x)_{x\in S}$ et le treillis $((J_x)_{x\in S}, \subseteq)$ portent ainsi une propriété de minimalité importante, et une famille de

Moore, ou encore un treillis de fermés, seront dit minimaux lorsque la cardinalité de l'ensemble sur lequel ils sont définis est égale au nombre des sup-irréductibles du treillis.

Notons cependant l'isomorphisme avec le treillis de fermés inversé défini sur l'ensemble des inf-irréductibles qui lui peut avoir une cardinalité inférieure :

$$(S, \leq) \cong ((J_x)_{x \in S}, \subseteq) \cong ((M_x)_{x \in S}, \supseteq) \tag{2.15}$$

Exemple 4 *Les deux treillis de fermés de la figure 2.7 sont isomorphes, et différents de par la cardinalité de l'ensemble sur lequel ils sont définis. Ils possèdent* 4 *sup-irréductibles. Ainsi, le premier est minimal car il est défini sur un ensemble de cardinalité* 4*, alors que le second ne l'est pas.*



(a) Treillis de fermés sur $\{a, b, c, d\}$

(b) Treillis de fermés sur $\{a, b, c, d, e\}$

FIGURE 2.7 – Treillis de fermés isomorphes

2.3.2 Système de fermeture

La propriété principale des treillis de fermés, et plus particulièrement de la famille qui les définit, réside dans les liens qui les unissent aux *systèmes de fermeture* :

Définition 9 (Système de fermeture) *Un* système de fermeture *est une paire* $C = (S, \varphi)$ *avec* S *un ensemble, et* φ *un* opérateur de fermeture*, i.e. une application définie sur* $\mathcal{P}(S)$ *à la fois isotone, extensive et idempotente :*

- *isotonie :* $X \subseteq Y$ *implique* $\varphi(X) \subseteq \varphi(Y)$
- *extensivité :* $X \subseteq \varphi(X)$
- *idempotence :* $\varphi(\varphi(X)) = \varphi(X)$

Pour une partie $X \subseteq S$ quelconque, $\varphi(X)$ est appelée la fermeture *de X, ou encore un* fermé *de C. L'ensemble des fermés forme une famille de Moore $\mathcal{F}_C$:*

$$\mathcal{F}_C = \{\varphi(X) \ : \ X \subseteq S\} \tag{2.16}$$

Dans la littérature, le terme *système de fermeture* est utilisé à la fois pour désigner la couple $C = (S, \varphi)$, mais aussi la famille des fermés $\mathcal{F}_C$ définie sur S. Le terme anglais *closure system* est également utilisé dans les deux situations, alors que le terme *closure space* est plus usité pour désigner le couple (S, φ).

La famille des fermés $\mathcal{F}_C$ possédant la propriété d'une famille de Moore, à tout système de fermeture C on associe son treillis de fermés $(\mathcal{F}, \subseteq)$. Dualement, à tout treillis de fermés $(\mathcal{F}, \subseteq)$ sur S on associe un système de fermeture dont l'opérateur de fermeture φ est défini, pour $X \subseteq S$, par :

$$\varphi(X) = \bigcap \{F \in \mathcal{F} \ : \ X \subseteq F\} \tag{2.17}$$

Les deux bornes $\vee$ et $\wedge$ se redéfinissent alors via l'opérateur de fermeture, pour X et Y parties de S, par :

$$\varphi(X) \wedge \varphi(Y) = \varphi(X) \cap \varphi(Y) \tag{2.18}$$

$$\varphi(X) \vee \varphi(Y) = \varphi(\varphi(X) \cup \varphi(Y)) \tag{2.19}$$

$$\tag{2.20}$$

La notion de générateur minimal d'un élément d'un treillis introduite dans la section 2.2.2 se redéfinit également via l'opérateur de fermeture : B est une base ou un générateur minimal pour un fermé F si $\varphi(B) = F$ et $\varphi(A) \subset \varphi(B)$ pour tout $A \subset B$.

Cette bijection entre treillis de fermés et systèmes de fermeture, ainsi que la bijection 1 entre treillis et treillis de fermés permettent d'étendre le cadre applicatif des treillis à celui des systèmes de fermeture :

Bijection 2 *Treillis et systèmes de fermeture sont en correspondance bijective.*

Plus précisément, à tout treillis $L = (S, \leq)$ on associe le système de fermeture $C = (J_L, \varphi)$ ou, pour $x \in S$, $\varphi(x) = J_x$. La minimalité de la famille de Moore sous-jacente $(J_x)_{x \in S}$ permet alors de définir la notion de *système de fermeture réduit* qui intègre à la fois la définition de l'opérateur de fermeture φ ainsi que celle d'un sup-irréductible.

Définition 10 (Système de fermeture réduit) *Un système de fermeture $C = (S, \varphi)$ est* réduit *lorsque l'ensemble S sur lequel il est défini ne contient que des éléments dont les fermés sont des sup-irréductibles de son treillis de fermés :*

$$\forall x \in S, \ \forall Y \subseteq S \text{ tel que } x \notin Y \quad \text{on a } \varphi(x) \neq \varphi(Y) \tag{2.21}$$

Si un système de fermeture n'est pas réduit, alors il contient au moins un élément réductible $x \in S$ dont le fermé $\varphi(x)$ n'est pas un sup-irréductible, i.e. qui ne vérifie pas l'équation 2.21. Il existe alors un sous-ensemble E_x d'éléments équivalents dont $\varphi(x)$ est la borne supérieure :

$$\varphi(x) = \varphi(E_x) = \bigvee_{y \in E_x} \varphi(y) \tag{2.22}$$

Notons que l'ensemble E_x est vide pour l'élément réductible $x \in S$ tel que $\varphi(x) = \varphi(\emptyset)$. Le fermé $\varphi(x)$ correspond alors à l'élément minimal du treillis.

La suppression des éléments réductibles d'un système de fermeture n'affecte pas la structure de son treillis de fermés. Par conséquent, la réduction d'un système de fermeture consiste à supprimer ou à remplacer chacun de ces éléments par l'ensemble E_x d'éléments qui lui sont équivalents.

Exemple 5 *Les deux treillis de fermés de la figure 2.7 sont isomorphes. Alors que le premier treillis de fermés est minimal - car $\varphi(x)$ est un sup-irréductible pour tout $x \in S$ - le second ne l'est pas. En effet, $\varphi(e)$ n'est pas un sup-irréductible. L'élément e est alors équivalent à $E_e = \{a, b\}$ car $\varphi(e) = \varphi(ab) = \varphi(a) \vee \varphi(b)$.*

Chapitre 3

Deux usages classiques d'un système de fermeture

On retrouve des opérateurs de fermeture dans de nombreux domaines, que ce soit en logique, en topologie, en bases de données, en combinatoire, ou encore en analyse de données. En particulier lorsque les données s'organisent sous forme d'une table binaire, appelée *contexte*, ou d'un système de *règles d'implications* du type "si" "alors". Dans ce chapitre, nous avons choisi de présenter le *treillis des concepts* d'un contexte, et le treillis de fermés d'un système d'implication comme deux cas d'usage des systèmes de fermeture, car leur existence repose sur celle d'opérateurs de fermeture.

En effet, la *connexion de Galois* entre les objets et les attributs qui définit tout contexte induit l'existence de deux opérateurs de fermeture, le premier défini sur l'espace des objets et le second sur l'espace des attributs. Le treillis des concepts correspond à la combinaison de deux treillis de fermés, le premier défini sur l'ensemble des attributs, le second sur l'ensemble des objets. Par ailleurs, un mécanisme d'inférence se définit naturellement par un opérateur de fermeture pour tout ensemble de règles d'implication. Son treillis de fermés est alors composé de toutes les parties qui vérifient l'ensemble des règles, appelées des *fermés*.

Ce choix, qui pose en premier plan les systèmes de fermeture, est également motivé par des considérations algorithmiques. En effet, les problèmes algorithmiques sous-jacents exploitent les propriétés de l'opérateur de fermeture.

La section 3.1 définit la notion de *contexte*, de *treillis des concepts* (section 3.1.1), et de *treillis de Galois* (section 3.1.2). Le treillis de Galois est présenté comme une généralisation du treillis des concepts à des données plus complexes pour lesquelles il existe une connexion de Galois.

La section 3.2 introduit quant à elle les *systèmes implicatifs* (section 3.2.1), ainsi que deux bases canoniques de règles d'implication, la directe et la non directe (section 3.2.2).

3.1 Treillis des concepts et treillis de Galois

Treillis de Galois et *treillis des concepts* sont des structures équivalentes qui diffèrent de par leur définition et l'usage qui en est fait. Le treillis des concepts, introduit dans [82], met en avant les notions d'objets et d'attributs du contexte à partir duquel il se définit, se positionnant ainsi dans le cadre de l'Analyse Formelle des Concepts (AFC) [58]. Le treillis de Galois quant à lui a été introduit dans les années 1970 dans [45]. Il se définit à partir d'une connexion de Galois entre deux ensembles, ce qui permet de retrouver les propriétés déjà mentionnées d'un système de fermeture (Section 2.3), mais aussi de positionner la structure de treillis dans un cadre plus large.

3.1.1 Treillis des concepts

En analyse de données, l'analyse formelle des concepts [58] pose le *treillis des concepts* comme espace de recherche sous-jacent à des données qui s'organisent sous forme d'une table binaire objets $\times$ attributs encore appelé *contexte*. Le treillis des concepts est un treillis dont les noeuds, appelés *concepts* sont composés à la fois d'un ensemble d'objets et d'un ensemble d'attributs.

Définition 11 (Treillis des concepts) *Le* treillis des concepts *se définit pour une relation binaire R entre un ensemble O d'objets et un ensemble I d'attributs, encore appelé* contexte. *Le treillis des concepts d'un contexte (O, I, R) est une paire $(\mathcal{C}, \trianglelefteq)$ où :*

– *$\mathcal{C}$ est un ensemble de* concepts *défini sur $\mathcal{P}(O) \times \mathcal{P}(I)$ par :*

$$(A, B) \in \mathcal{C} \iff A \subseteq O, B \subseteq I, B = \alpha(A) \text{ et } A = \beta(B)$$

avec

$$\begin{aligned} \alpha(A) &= \{b \in I \ : \ aRb \text{ pour tout } a \in A\} \\ \beta(B) &= \{a \in O \ : \ aRb \text{ pour tout } b \in B\} \end{aligned}$$

– *$\trianglelefteq$ est une relation binaire définie sur l'ensemble des concepts $\mathcal{C}$ par, pour (A_1, B_1) et $(A_2, B_2) \in \mathcal{C}$:*

$$(A_1, B_1) \trianglelefteq (A_2, B_2) \iff B_1 \subseteq B_2 \iff A_1 \supseteq A_2 \tag{3.1}$$

Lorsque le contexte est représenté sous forme d'une table, un concept correspond à un rectangle maximal, ou encore à une sous-matrice première. Un contexte pourra être noté (O, I, R) ou encore $(O, I, (\alpha, \beta))$ selon que l'on considère la relation binaire R entre O et I, ou, de façon équivalente, les deux applications α et β, l'une de O vers I, l'autre de I vers O.

Dans [58], l'ensemble des objets est noté G pour "Gergenstand" et l'ensemble des attributs M pour "Merkmalle", et un contexte est noté (G, M, R). Un seul opérateur

	a	b	c	d	e	f	g	h
1		×					×	
2	×		×				×	
3				×	×	×	×	×
4	×			×	×	×	×	×
5	×				×	×	×	×
6		×				×	×	
7	×					×	×	
8	×				×		×	
9	×	×	×	×	×	×	×	×

TABLE 3.1 – Exemple de contexte

"prime" est utilisé à la place des opérateurs α et β, et un concept (A, B) y est défini par $A \subseteq G$, $B \subseteq M$, $B = A'$ et $A = B'$.

Les bornes supérieure et inférieure dans un treillis des concepts sont définies, pour deux concepts (A_1, B_1) et (A_2, B_2) par :

$$(A_1, B_1) \wedge (A_2, B_2) = (\beta(B_1 \cap B_2), (B_1 \cap B_2)) \tag{3.2}$$

$$(A_1, B_1) \vee (A_2, B_2) = ((A_1 \cap A_2), \alpha(A_1 \cap A_2)) \tag{3.3}$$

Exemple 6 *Considérons à titre d'exemple le contexte décrit par la table 3.1 définie par un ensemble d'objets* $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ *et un ensemble d'attributs* $\{a, b, c, d, e, f, g, h\}$*, et son treillis des concepts donné Figure 3.1. On remarque que :*

- *l'objet numéro 9 possède tous les attributs, et par conséquent apparaît dans tous les concepts.*
- *l'attribut g est partagé par tous les objets, apparaissant également dans tous les concepts.*
- *l'attribut h est partagé par les objets* 3, 4, 5 *et* 9*, i.e. par les objets possédant à la fois l'attribut e et l'attribut f. Par conséquent, l'attribut h n'apparaît que dans les concepts contenant à la fois l'attribut e et l'attribut f.*

3.1.2 Treillis de Galois

Le treillis de Galois se définit à partir d'une correspondance de Galois entre deux ensembles qui elle-même définit deux opérateurs de fermetures sur chacun des deux ensembles, permettant ainsi d'étendre aux treillis de Galois les propriétés des systèmes de fermeture.

Définition 12 (Treillis de Galois) *Un* treillis de Galois *se définit à partir d'une* correspondance de Galois (α, β) *entre deux ensembles S et U où :*

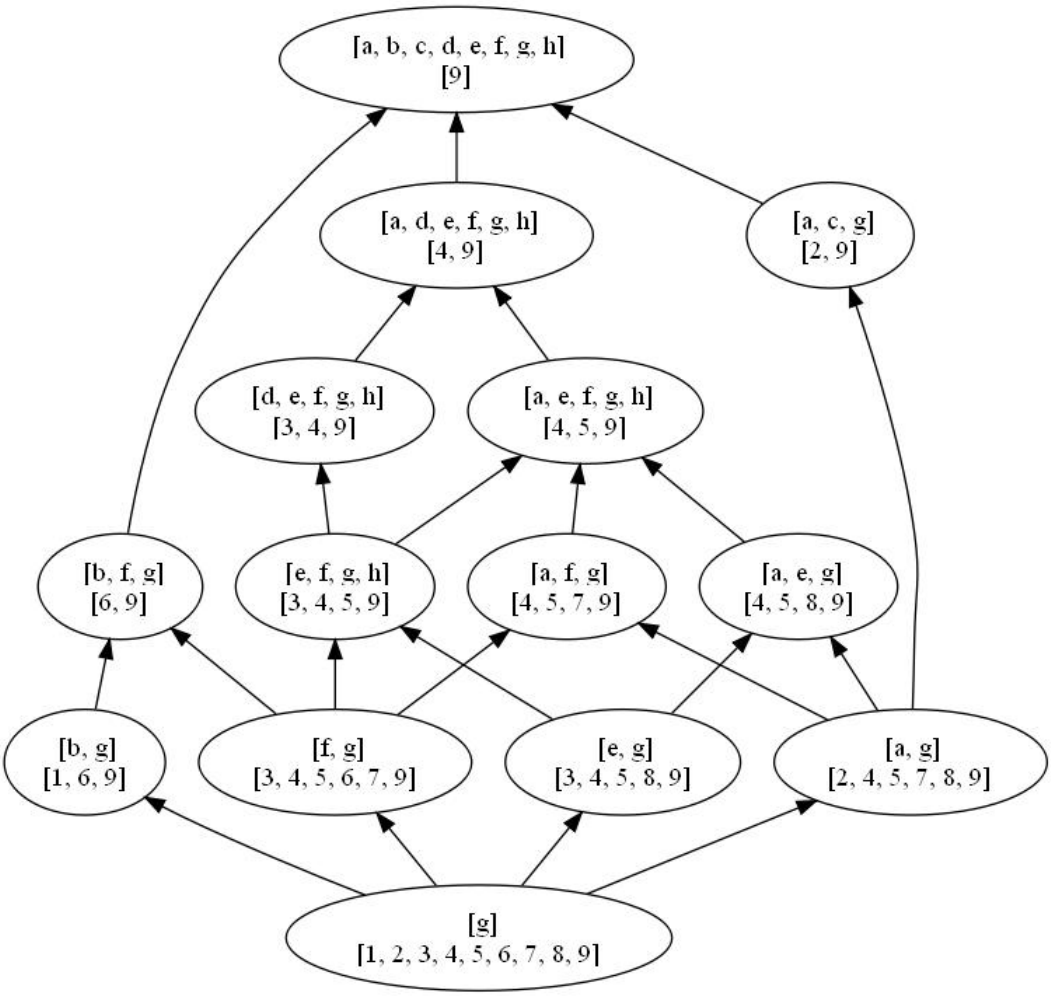


FIGURE 3.1 – Treillis des concepts du contexte de la table 3.1

- *α est une application isotone de $\mathcal{P}(S)$ vers $\mathcal{P}(U)$: $X \subseteq Y$ implique $\alpha(X) \subseteq \alpha(Y)$,*
- *β est une application antitone de $\mathcal{P}(U)$ vers $\mathcal{P}(S)$: $X \subseteq Y$ implique $\beta(X) \supseteq \beta(Y)$,*
- *$(\beta \circ \alpha)$ est une application extensive sur $\mathcal{P}(U)$: $X \subseteq U$ implique $X \subseteq (\beta \circ \alpha)(X)$,*
- *$(\alpha \circ \beta)$ est une application extensive sur $\mathcal{P}(S)$: $X \subseteq S$ implique $X \subseteq (\alpha \circ \beta)(X)$.*

En d'autres termes, $(\alpha \circ \beta)$ et $(\beta \circ \alpha)$ sont deux opérateurs de fermeture - le premier défini sur S et le second défini sur U - dont les deux treillis de fermés sont isomorphes et forment le treillis de Galois $(\mathcal{C}, \trianglelefteq)$ de la correspondance de Galois, avec $\trianglelefteq$ relation binaire définie sur $\mathcal{C}$ par l'équation 3.1, et $\mathcal{C}$ ensemble défini sur $\mathcal{P}(S) \times \mathcal{P}(S)$ par :

- *$(F, \alpha(F)) \in \mathcal{C}$, pour tout fermé F du système de fermeture $(S, (\beta \circ \alpha))$* [1].
- *$(\beta(F), F) \in \mathcal{C}$ pour tout fermé F du système de fermeture $(U, \alpha \circ \beta)$* [2].

Par ailleurs, la définition du treillis des concepts d'un contexte (O, I, R) introduit une paire (α, β) qui forme une correspondance de Galois entre les deux ensembles $\mathcal{P}(O)$ et $\mathcal{P}(I)$. La structure de treillis du treillis des concepts est ainsi une conséquence directe de la correspondance de Galois portée par tout contexte, et les notions de fermeture et de fermés s'y retrouvent tout naturellement. En effet, à tout contexte $(O, I, (\alpha, \beta))$ on associe les deux systèmes de fermeture $(O, \alpha \circ \beta)$ et $(I, \beta \circ \alpha)$, et les deux treillis de fermés sous-jacents. Ainsi, tout treillis des concepts correspond à l'imbrication de ces deux treillis de fermés, le premier défini sur l'ensemble des objets, le second sur l'ensemble des attributs.

Cette correspondance entre treillis des Galois et treillis de fermés s'étend ainsi tout naturellement aux treillis des concepts. Elle permet de manipuler de façon équivalente treillis et treillis des concepts sur la base des éléments irréductibles du treillis :

Bijection 3 *[45] Tout treillis est isomorphe au treillis des concepts de sa table binaire (cf Définition 6).*

Cette table définit un contexte dont les éléments sup-irréductibles (resp. inf-irréductibles) sont les attributs (resp. les objets). Plus précisément, à tout élément $x \in S$ d'un treillis $(S, \leq)$ on associe la paire (J_x, M_x) composée d'un sous-ensemble de sup-irréductibles et d'un sous-ensemble d'inf-irréductibles. On vérifie facilement que cette paire forme un concept de la table du treillis, et on a :

$$(S, \leq) \cong ((J_x, M_x)_{x \in S}, \trianglelefteq) \tag{3.4}$$

Un contexte est dit *réduit* lorsque les deux treillis de fermés qui composent son treillis des concepts sont minimaux, i.e. définis pour deux systèmes de fermeture réduits dont la définition (cf définition 10) permet d'en caractériser les attributs comme des sup-irréductibles et les objets comme des inf-irréductibles. Un *contexte réduit* est alors

1. $\alpha(F)$ est alors un fermé de $(U, \alpha \circ \beta)$
2. $\beta(F)$ est alors un fermé de $(S, \beta \circ \alpha)$

un contexte dont les attributs et les objets sont irréductibles. Un attribut est dit sup-irréductible si sa fermeture selon $\beta \circ \alpha$ est un sup-irréductible, et ne contient pas d'autres attributs avec la même fermeture. Cette fermeture se retrouve dans le plus petit concept contenant l'attribut. Dualement, un objet est dit inf-irréductible lorsque sa fermeture selon $\alpha \circ \beta$ est un inf-irréductible, et ne contient pas d'autres objets de même fermeture. On retrouve cette fermeture dans le plus grand concept contenant l'objet.

La table d'un treillis définie par ses sup-irréductibles en colonnes et ses inf-irréductibles en ligne possède tout naturellement cette propriété de réduction :

Bijection 4 *[66, 67] Tout contexte réduit est isomorphe à la table binaire de son treillis des concepts.*

Lorsqu'un contexte n'est pas réduit, il est possible de le réduire en supprimant ses éléments réductibles que sont les attributs et objets ne vérifiant pas l'équation 2.21 de la définition 10 d'un système de fermeture réduit. La structure du treillis des concepts n'en sera pas affectée. En particulier, un attribut a partagé par tous les objets sera supprimé car, pour $X = \emptyset$, $(\beta \circ \alpha)(a) = (\beta \circ \alpha)(\emptyset)$.

Exemple 7 *On peut remarquer que les treillis des concepts des figures 2.3 et 3.1 sont isomorphes. Celui de la figure 3.1 est le treillis des concepts du contexte non réduit de la table 3.1. Celui de la Figure 2.3 est quant à lui composé des concepts $(J_x, M_x)_{x \in S}$ de la table binaire (table 2.2) du treillis de la figure 2.1(b). La table binaire correspond à la réduction du contexte d'où ont été supprimés les éléments suivants :*

- *L'objet 9 du contexte possède tous les attributs, et par conséquent apparaît dans tous les concepts du treillis. Son concept associé, élément maximal du treillis, n'est pas un inf-irréductible, et il sera pas conséquent supprimé par réduction. En effet, il ne vérifie pas l'équation 2.21, et il n'existe pas dans la table binaire d'objet possédant tous les attributs.*
- *L'attribut g du contexte est partagé par tous les objets, et par conséquent apparaît dans tous les concepts du treillis. Il n'est pas irréductible et sera supprimé par réduction. On peut vérifier qu'il ne vérifie pas l'équation 2.21 et qu'il n'apparaît pas dans la table binaire.*
- *L'attribut h du contexte est partagé par les objets 3, 4, 5 et 9, i.e. par tous les objets possédant à la fois les attributs e et f. L'attribut h apparaît ainsi dans tout concept contenant à la fois les attributs e et f, et son concept associé, le concept $(efgh, 3459)$ n'est pas un sup-irréductible. On peut vérifier que l'attribut h ne vérifie pas l'équation 2.21 et qu'il n'apparaît pas dans la table binaire du treillis (table 2.2).*

Alors que la définition du treillis des concepts utilise une description ensembliste des objets O du contexte - tout objet est décrit par un ensemble d'attributs - la définition d'un treillis de Galois permet d'envisager une description plus sophistiquée des objets

par un ensemble de descriptions $\mathcal{D}$ à condition de maintenir les propriétés de l'application α qui sont garanties lorsqu'il existe une relation de subsomption $\sqsubseteq$ ou une relation d'inclusion $\sqcap$, l'une se déduisant de l'autre :

$$c \sqsubset d \iff c \sqcap d = c \quad \forall c, d \in \mathcal{D} \tag{3.5}$$

La correspondance de Galois entre les objets $(O, \subseteq)$ et leurs descriptions $(\mathcal{D}, \sqsubseteq)$ est alors maintenue, garantissant ainsi l'existence d'un treillis de Galois pour $(O, \mathcal{D})$.

Dans [75, 64], le treillis de Galois est ainsi utilisé pour des objets décrits par des intervalles, des histogrammes ou encore des données multivaluées pour lesquels une relation de subsomption est définie. L'*Analyse Logique des Concepts* [55] a été introduite en 2004 comme une généralisation de l'AFC où le treillis de Galois est défini à partir d'un *contexte logique* contenant une description de chaque objet par une formule logique. Le formalisme des *structures de patrons*, introduit dans [57] pour des descriptions d'objets par des graphes, également étudiée dans [64] pour des données numériques, est une autre généralisation de ce même principe.

3.2 Systèmes implicatifs

En logique, ou encore en fouille de données, les *systèmes implicatifs*, ou systèmes de règles d'implication, sont utilisés pour exprimer des implications entre des données. Le mécanisme d'inférence associé, ou mécanisme de chaînage avant, est classiquement défini par un opérateur de fermeture, ce qui permet de positionner les systèmes implicatifs comme des systèmes de fermeture, et ainsi d'étendre le champ applicatif des treillis au règles d'implication [50]. La section 3.2.1 pose les principales définitions : systèmes d'implication, mais aussi opérateur de fermeture associé, et principales propriétés. Plusieurs systèmes implicatifs différents peuvent engendrer le même treillis de fermés via l'opérateur de fermeture, ce qui permet de considérer certaines propriétés, ainsi que des mécanismes de transformation pour les obtenir. La section 3.2.2 introduit deux bases de systèmes implicatifs avec une propriété de minimalité : la *base canonique* et la *base canonique directe*.

3.2.1 Définition d'un système implicatif

Définition 13 (Système implicatif) *Un* système implicatif *(ou SI)* Σ *sur un ensemble d'éléments* S *est une relation binaire définie sur* $\mathcal{P}(S)$*. Une* règle d'implication *est un couple* $(A, B) \in \Sigma$ *notée* $A \to B$ *et signifiant* A implique B*.* A *est la* prémisse *de la règle, et* B *sa* conclusion*.*

A tout système implicatif Σ, on associe la famille des parties de S fermées pour les règles d'implication :

$$\mathcal{F}_\Sigma = \{X \subseteq S \ : \ A \subseteq X \text{ et } A \to B \text{ impliquent } B \subseteq X \text{ pour tout } A \to B \in \Sigma\} \tag{3.6}$$

On montre facilement que cette famille possède les propriétés d'une famille de Moore, à savoir stabilité par intersection, et appartenance de l'ensemble S lui-même (cf Section 2.3). Par conséquent, ceci implique à la fois que $\mathcal{F}_\Sigma$ muni de la relation d'inclusion forme un treillis, mais également que $\mathcal{F}_\Sigma$ peut étre générée à partir d'un opérateur de fermeture φ_Σ qui définit la fermeture d'un sous-ensemble X de S par :

$$\varphi_\Sigma(X) = \pi(X) \cup \pi^2(X) \cup \pi^3(X) \cup \dots \text{ avec} \tag{3.7}$$

$$\pi(X) = X \cup \bigcup \{B \mid A \subseteq X \text{ et } A \to B \in \Sigma\} \tag{3.8}$$

Plusieurs itérations - au plus $|S|$ itérations dans le pire des cas - sont donc nécessaires pour calculer une fermeture. En cas de non ambigüité, on notera φ au lieu de φ_Σ.

Deux systèmes implicatifs différents Σ et Σ' peuvent engendrer deux familles de Moore isomorphes - i.e. $\mathcal{F}_\Sigma \cong \mathcal{F}_{\Sigma'}$. Un système implicatif est dit *réduit* lorsque la famille de Moore qu'il engendre est minimale - i.e. la cardinalité de l'ensemble S qui la définit est minimale - S correspond alors aux sup-irréductibles du treillis de fermés. Il est par conséquent possible de réduire un système implicatif en remplaçant dans chaque règle chaque élément réductible par ses irréductibles équivalents (cf Définition 10).

Deux systèmes implicatifs Σ et Σ' sont dits *equivalents* lorsqu'ils engendrent deux familles de Moore identiques - i.e. $\mathcal{F}_\Sigma = \mathcal{F}_{\Sigma'}$. Parmi l'ensemble des systèmes implicatifs équivalents, certains se distinguent par certaines propriétés.

Un système implicatif Σ est dit *unaire* lorsque chacune de ses règles possède un singleton en conclusion. On peut alors définir Σ sur $\mathcal{P}(S) \times S$, et écrire $B \to x$ au lieu de $B \to \{x\}$, avec $B \subseteq S$ et $x \in S$. Rendre unaire un système implicatif Σ consiste alors à remplacer toute règle $B \to X = \{x_1, \dots, x_n\} \in \Sigma$ par les n règles unaires $B \to x_i$, pour $i \leq n$. À l'inverse, un système implicatif est dit *compact* lorsque chacune de ses règles est composée d'une conclusion de cardinalité maximale, ou, de façon équivalente, s'il n'existe pas deux règles distinctes de même prémisse. Rendre compact un système implicatif Σ consiste alors à remplacer les n règles de même prémisse B par une seule règle dont la prémisse est B, et la conclusion est l'union des conclusions des n règles. Un système implicatif est *propre* lorsque la prémisse et la conclusion de chacune de ses règles ne s'intersectent pas. Pour rendre propre un système implicatif Σ, il suffit de remplacer chaque règle $B \to X$ par la règle propre $B \to X \setminus B$. Un système implicatif est dit *maximal à droite* lorsqu'il est compact, et que la conclusion de chacune de ses règles de prémisse X est égale à $\varphi(X) \setminus X$. Un système implicatif est dit *minimal à gauche* lorsqu'il est compact, et qu'il n'existe pas deux règles de même prémisse.

Un système implicatif est *direct* si la fermeture de toute partie $X \subseteq S$ s'obtient en appliquant l'équation 3.8 une seule fois, i.e. $\varphi(X) = \pi(X)$. Un système implicatif Σ est *minimal* ou *non redondant* si aucune règle ne peut être supprimée, i.e. pour toute règle $X \to Y \in \Sigma$, $\Sigma \setminus \{X \to Y\}$ n'est pas équivalent à Σ. Un système implicatif Σ est *minimum* s'il est de cardinalité minimale, i.e. si $|\Sigma| \leq |\Sigma'|$ pour tout système d'implication Σ' équivalent à Σ. Un système implicatif minimum est clairement non

redondant, cependant, l'inverse n'est pas vérifié. Un système implicatif minimum est également appelé une *base* pour le système de fermeture qu'il engendre, ainsi que pour le treillis des fermés induit. Une *base minimale* est ainsi une base de cardinalité minimale. Deux bases se distinguent : la *base canonique*, et la *base canonique directe*.

3.2.2 Base canonique et base canonique directe

La *base canonique* Σ_{cb}, également appelée la *base de Guigues et Duquenne* [60] est l'unique base minimum à partir de laquelle toutes les autres règles d'implications peuvent s'inférer en utilisant les axiomes de Amstrong [44]. Elle se défnit par des règles dont les prémisses sont des *pseudo-fermés* :

Définition 14 (Base canonique) *[60] La* base canonique *sur S est définie par :*

$$\Sigma_{cb} = \{B \rightarrow \varphi(B) \setminus B, : B \subseteq S \text{ est un pseudo-fermé de } \mathcal{F}_\Sigma\} \tag{3.9}$$

où un pseudo-fermé est une partie $B \subseteq S$ telle que $B \notin \mathcal{F}_\Sigma$, $\mathcal{F} + B$ est une famille de Moore, et B minimal par inclusion pour cette propriété.

La famille de tous les pseudo-fermés de $\mathcal{F}_\Sigma$ sur S ainsi définis forme également une famille de Moore, mais non isomorphe à $\mathcal{F}_\Sigma$.

Exemple 8 *Les deux treillis de fermés de la figure 2.7 sont isomorphes, le premier est minimal, le second ne l'est pas car $\varphi(e)$ n'est pas un sup-irréductible, l'élément e est alors équivalent à $E_e = \{a, b\}$. La base canonique du premier treillis de fermés est définie sur l'ensemble $S = \{a, b, c, d\}$ par $\Sigma_{cb} = \{c \rightarrow a \ ; \ d \rightarrow b\}$. La base canonique du second treillis de fermés est quant à elle définie sur l'ensemble $S = \{a, b, c, d, e\}$ par $\Sigma_{cb} = \{c \rightarrow a \ ; \ d \rightarrow b \ ; \ ab \rightarrow e\}$. La réduction du système implicatif de ce deuxième treillis consiste alors à remplacer e par ab dans chaque règle, ce qui revient à supprimer la troisième règle.*

Nous avons introduit la base canonique directe Σ_{cdb} *comme l'unique base minimale parmi l'ensemble des systèmes implicatifs équivalents et directs. Cette base étant directe, ceci signifie que le mécanisme d'inférence induit par φ est simplifié (voir Equation 3.7) : chaque fermeture est alors définie par $\varphi(X) = \pi(X)$, et peut par conséquent s'obtenir en considérant chaque règle de la base canonique directe une seule fois. Rappelons qu'avec un système implicatif non direct, et en particulier avec la base canonique, plusieurs itérations sur l'ensemble des règles sont nécessaires pour calculer une fermeture. Notons cependant que la base canonique directe a un nombre de règles d'implications bien supérieur à celui de la base canonique.*

Alors qu'il existe une unique définition de la base canonique, on retrouve la base canonique directe sous diverses définitions issues de domaines différents. Dans de récents travaux [11], nous avons mis en évidence que cette base est équivalente à cinq

autres bases de la littérature, permettant ainsi de regrouper les définitions et propriétés de chacune d'entre elles, définies sous forme unaire, i.e. $\Sigma_{cdb} \in \mathcal{P}(S) \times S$ *:*

La *base directe-optimale* [16]. *Nous avons introduit cette base à l'aide d'un algorithme de génération à partir d'un système implicatif équivalent propre et unaire* Σ *:*

1. *application récursive du traitement suivant pour obtenir un système implicatif équivalent, unaire et direct :*[3] *:*

   pour tout $A \to b$ *et* $C + b \to d$ *de* Σ*, ajouter* $A \cup C \to d$ *dans* Σ

2. *puis application du traitement de minimalisation suivant :*

   pour tout $A \to b$ *et* $C \to b$ *de* Σ*, si* $C \subset A$ *alors supprimer* $A \to b$ *de* Σ*.*

Cet algorithme de génération nous a permis de montrer la minimalité de la base directe-optimale parmi les bases directes, et ainsi d'en déduire son unicité. Cette définition introduit un traitement permettant de rendre direct un système implicatif quelconque, traitement que l'on retrouve en bases de données sous la dénomination de pseudo-transitivité ou encore d'opérateur d'accumulation [65].

La *base associée à la relation de dépendance.* *Il s'agit là de la base que nous avons définie à partir du graphe de dépendance et de la relation de dépendance entre irréductibles d'un treillis, introduite en 1990 [68] :*

$$\Sigma_\delta = \{X + y \to x \ : \ x\delta_X y \text{ et } X \text{ est minimal par inclusion pour cette propriété}\} \tag{3.10}$$

où la relation de dépendance δ_X *est définie sur* S*, pour* $x, y \in S$ *et* $X \subset S$*, par :*

$$x\delta_X y \text{ si et seulement si } x \notin \varphi(X),\ y \notin \varphi(X) \text{ et } x \in \varphi(X + y) \tag{3.11}$$

La définition de la relation δ_X *est similaire à celle utilisée pour introduire le graphe de dépendance (définition 7), permettant ainsi de faire le lien avec les générateurs minimaux du treillis.*

La *base canonique et "iteration-free"* [81]. Introduite en 1994, elle se définit à partir de la notion de sous-ensemble libre :

$$\Sigma_{cif} = \{B \to x \ : \ x \in \varphi(B) \backslash \pi_\varphi(B) \text{ et } B \text{ est un sous-ensemble libre}\} \tag{3.12}$$

où π_φ est défini à partir de φ par[4] :

$$\pi_\varphi(B) = B \cup \{x \in S \ : \ \text{il existe } A \subset B \text{ avec } x \in \varphi(A)\} \tag{3.13}$$

Un sous-ensemble B de S est *libre* si, pour tout $x \in B$ $x \notin \varphi(B \backslash x)$. De façon équivalente, B est libre si et seulement si $\varphi(A) \subset \varphi(B)$ pour tout $A \subset B$. Ainsi, B est libre si c'est un générateur minimal de $\varphi(B)$.

3. Si Σ n'est pas propre, il est nécessaire d'ajouter la condition $b \notin A$, $b \neq d$ et $d \notin A \cup C$.
4. Si B n'est pas un ensemble libre, il est nécessaire d'ajouter la condition $\varphi(A) \subset \varphi(B)$.

La *base minimale à gauche.* Elle se définit par des règles dont la prémisse est de cardinalité minimale :

$$\Sigma_{lm} = \{X \to y \ : \ y \in \varphi(X) \setminus X \text{ et pour tout } X' \subset X,\ y \notin \varphi(X')\} \tag{3.14}$$

Une implication de la base minimale à gauche est appelée une *dépendance fonctionnelle minimale* dans le domaine des bases de données relationnelles ([65]), ou encore une *implication propre* dans [80], où elles sont utilisées dans un contexte de fouille de données, avec des prémisses appelées *générateurs minimaux* du treillis.

La *base faible d'implication.* Introduite en 1995 [77], elle est définie à partir des transversales minimales d'une famille de parties appelées *copoints* :

$$\Sigma_{weak} = \{B \to x \ : \ B \subseteq S \text{ et } B \text{ est un ensemble bloquant pour } x\} \tag{3.15}$$

où un *ensemble bloquant* pour $x \in S$ est une transversale minimale de la famille $\mathcal{D}_x$ définie sur S par :

$$\mathcal{D}_x = \{S \backslash (C + x)\,,\ C \text{ est un copoint de } x\} \tag{3.16}$$

Un sous-ensemble C de S est un *copoint* de $x \in S$ si C est un sous-ensemble maximal de S tel que $x \notin \varphi(C)$. Il a été établi que les copoints de x sont les inf-irréductibles du treillis de fermés $(\mathcal{F}_\varphi, \subseteq)$. Cette définition a permit d'établir une connexion avec les espaces de connaissance ([54]), familles de parties stables par union et contenant l'ensemble vide.

Exemple 9 *La base canonique du treillis de la figure 2.1(b) est définie sur son ensemble de sup-irréductibles* $\{a, b, c, d, e, f\}$ *par :*

$$\Sigma_{cb} = \{d \to ef\ ;\ c \to a\ ;\ be \to acdf\ ;\ acf \to bde\ ;\ ace \to bdf\ ;\ ab \to cdef\}$$

Sa base canonique directe est définie par :

$$\Sigma_{cdb} = \{d \to ef\ ;\ c \to a\ ;\ cf \to bde\ ;\ ce \to bdf\ ;$$
$$cd \to b\ ;\ be \to acd\ ;\ f\ ;\ bd \to ac\ ;\ bc \to def\ ;\ ab \to cdef\}$$

Les trois dernières définitions introduisent les prémisses de la base canonique directe sous des terminologies différentes, tout en étant similaires : on parle d'ensemble libre pour la base iteration-free, d'ensemble bloquant pour la base faible d'implications, de générateur minimal pour la base des implications propres, ou encore de valuations du graphe de dépendance. La correspondance entre un ensemble bloquant et un générateur minimal, également établie dans [109], permet de proposer un calcul des générateurs minimaux par une génération de transversales minimales.

La définition de la base associée à la relation de dépendance introduit quant à elle une correspondance avec le graphe de dépendance d'un treillis (voir la définition 7). En effet, à tout arc (j, j') du graphe de dépendance dont la valuation contient le générateur minimal $X \subseteq J_L$ - i.e. $j \delta_X j'$ - correspond :

– la règle unaire $X + j' \to j$ de la base canonique directe et
– le générateur minimal $X + j'$ de $\varphi(X + j')$.

Cette correspondance implique deux propriétés fortes de la base canonique directe : une définition récursive des générateurs minimaux est reprise pour proposer un algorithme de calcul plus efficace que dans [109, 88], et le lien avec le graphe de dépendance d'un treillis permet d'établir les bijections suivantes :

Bijection 5 *Tout treillis est le treillis de fermés du système implicatif porté par le graphe de dépendance de son treillis de fermés.*

Bijection 6 *Toute base canonique directe et réduite est isomorphe au système implicatif porté par le graphe de dépendance de son treillis de fermés.*

Lien avec les fonctions booléennes

Une fonction booléenne f définie sur un ensemble S de variables - ou littéraux - peut s'écrire sous forme normale disjonctive (FND) ou sous forme normale conjonctive (FNC). Une forme normale est construite à partir des opérateurs de conjonction (*et logique*), de disjonction (*ou logique*), ainsi que l'opérateur de négation qui ne peut que précéder un littéral. On parle alors de littéral négatif ou positif selon qu'il est ou non précédé de l'opérateur de négation. La FND s'exprime comme une disjonction de *termes*, où un terme est une conjonction de littéraux. A l'inverse, la FNC s'exprime comme une conjonction de *clauses*, où une clause est une disjonction de littéraux.

La simplification d'une fonction booléenne en un nombre minimal de termes ou de clauses - selon qu'elle soit sous FND ou sous FNC - alors appelés *implicants premiers* est un problème NP-complet. A toute fonction booléenne f on associe la famille :

$$\mathcal{F}_f = \{X \subseteq S \ : \ f(X) = 1\} \tag{3.17}$$

Un fonction booléenne est une *fonction de Horn* lorsque, sous FND, les termes qui la composent ne contiennent qu'un seul littéral négatif, où de façon équivalente lorsque, sous FNC, elle est composée de *clauses* ne possédant qu'un seul littéral positif. On parle alors de *termes de Horn* ou de *clauses de Horn*.

Une autre définition moins connue d'une fonction de Horn utilise la notion de famille. Une fonction booléenne f est une fonction de Horn lorsque sa famille $\mathcal{F}_f$ est stable par intersection. Il s'agit alors, lorsqu'elle est pure, d'une famille de Moore qui, munie de la relation d'inclusion, forme un treillis.

On trouve dans [63] une correspondance entre dépendances fonctionnelles et implicants premiers d'une fonction booléenne, correspondance qui s'étend ainsi à la base canonique directe :

Bijection 7 *[63, 11] Les implicants premiers d'une fonction de Horn sont en correspondance bijective avec les règles de la base canonique directe.*

Plus précisément, à toute règle $B = \{b_1 \dots b_n\} \rightarrow x$ de la base canonique directe correspond le terme implicant premier $b_1 \dots b_n x'$ ou de façon équivalente la clause implicant premier $b'_1 + \dots + b'_n + x$.

Chapitre 4

Aspects algorithmiques

L'objet de ce chapitre est de fournir un panorama des différents algorithmes permettant de générer et manipuler les objets issus de la théorie d'un treillis. Instanciables en particulier par un contexte et un système implicatif, les systèmes de fermeture sont ici manipulés de façon générique pour une approche algorithmique commune (section 4.1). Des problèmes difficiles de génération sont ainsi abordés (c'est pourquoi quelques éléments de complexité sont définis en préalable dans la section 4.2) : génération des éléments irréductibles à partir d'un treillis ou d'un système de fermeture (section 4.3) ; génération du treillis de fermés d'un système de fermeture (section 4.4) ; génération de la base canonique d'un treillis ou d'un système de fermeture (section 4.5) ; génération de la base canonique directe d'un treillis ou d'un système de fermeture (section 4.6). Enfin, nous présentons dans la section 4.7 la bibliothèque *lattice* qui implémente ces algorithmes pour une manipulation efficace dans un contexte pédagogique ou applicatif expérimental.

4.1 Système de fermeture

La notion de système de fermeture (S, φ) (voir définition 9), ou tout au moins celle d'opérateur de fermeture φ, est une notion que l'on retrouve dans de nombreux domaines, sous des formes de représentations différentes. Deux usages classiques d'un système de fermeture ont été introduits dans la section 3 pour un contexte ou pour un système de règles d'implication. Bien qu'il existe des algorithmes efficaces pour ces deux structures, la donnée de l'opérateur φ et de l'ensemble S sur lequel il s'applique est suffisante pour une approche algorithmique plus générique et tout aussi efficace pour manipuler un système de fermeture quelconque : la génération des irréductibles, de l'ensemble des fermés ou encore à celle des deux bases canoniques peuvent ainsi être abordée de façon générique.

On définit à partir d'un contexte $(O, I, (\alpha, \beta))$ les deux systèmes de fermeture $(O, (\alpha \circ \beta))$ et $(I, (\beta \circ \alpha))$ - le premier défini sur l'ensemble des objets, le second sur l'ensemble des attributs. La fermeture $(\alpha \circ \beta)(X)$, avec $X \subseteq O$, ou encore la fermeture $(\beta \circ \alpha)(X)$

pour $X \subseteq I$ se calcule en $O(|O| \times |I|)$ par un simple parcours de la table selon α et selon β.

Un système implicatif n'engendre quant à lui qu'un seul système de fermeture (φ, S) (cf Equation 3.7). En revanche, on trouve dans la littérature plusieurs algorithmes de génération d'une fermeture engendrée par un système implicatif. L'algorithme *Linclosure* proposé dans [103] par Mannila et Räihä génère une fermeture $\varphi(X)$ en $O(|S|^2|\Sigma|)$ en parcourant itérativement l'ensemble des règles d'implication afin d'enrichir la fermeture, et ce jusqu'à ce qu'aucun nouvel élément ne lui soit ajouté (cf Algorithme 1). L'algorithme de Wild [116] (cf Algorithme 2) permet de limiter le nombre d'itérations nécessaires pour générer la fermeture de X en $O(|S||\Sigma|)$ en utilisant une structure de données plus sophistiquée qui permet d'accéder, pour tout élément $x_i nS$, à la liste des implications contenant x dans leurs prémisses.

Le temps de calcul dépend à la fois du nombre d'itérations nécessaires - borné par $|S|$ dans le pire les cas - mais aussi du nombre de règles $|\Sigma|$. Une transformation préalable de Σ en sa base canonique - base de cardinalité minimale - permet ainsi de minimaliser le nombre de règles (cf section 4.5).

Cependant, l'utilisation d'une base directe permet de minimaliser le nombre d'itérations car une seule itération est suffisante pour calculer chaque fermé. Nous exploitons cette propriété dans [16] en utilisant une base directe, et plus précisément la base canonique directe - base minimale parmi les bases directes - pour proposer la calcul de la fermeture de X en $O(|X||\Sigma| + |S|)$. Notons cependant que la taille de la base canonique directe est beaucoup plus grande, voire exponentielle dans le pire des cas, en celle de la base canonique.

Nom : `fermeture`
Données : Un système implicatif Σ défini sur S;
Un sous-ensemble X de S
Résultat : La fermeture $\varphi(X)$
début
  $Res = X$; $Prec = \emptyset$;
  while $Prec \neq Res$ **do**
    $Prec = Res$;
    foreach $A \rightarrow B \in \Sigma$ **do**
      si $A \subseteq Res$ **alors** ajouter B dans Res;
    end
  end
fin

Algorithme 1: Calcul d'une fermeture dans un système implicatif

En conclusion, le coût c_φ de génération de la fermeture de $X \subseteq S$ dépend avant tout du système de fermeture :

– pour un contexte $(O, I, (\alpha, \beta))$: $c_\varphi = O(|O| \times |I|)$;

Nom : `fermetureAvecCompteur`
Données : Un système implicatif Σ défini sur S;
Un sous-ensemble X de S
Résultat : La fermeture $\varphi(X)$
début
  foreach $x \in S$ **do**
    construire la liste L_x des implications contenant x dans la prémisse
  end
  $Res = X$;
  foreach $A \to B \in \Sigma$ **do**
    initialiser le compteur *cpt* de $A \to B$ à $|A|$
  end
  foreach $x \in Res$ *non marqué* **do**
    foreach $A \to B \in L_x$ **do**
      décrémenter le compteur *cpt* de $A \to B$
    end
    si $cpt = 0$ **alors** ajouter B dans Res;
    marquer x
  end
  retourner Res;
fin

Algorithme 2: Calcul d'une fermeture dans un système implicatif à l'aide d'un compteur

– pour un système d'implication Σ sur S non direct : $c_\varphi = O(|S||\Sigma|)$;
– pour un système d'implication Σ sur S direct : $c_\varphi = O(|X||\Sigma| + |S|)$.

4.2 Problèmes de génération et complexité

La famille $\mathcal{F}$ du treillis de fermés, ou encore la base canonique directe Σ_{cdb}, toutes deux définies pour un système de fermeture (S, φ), ont une taille au pire exponentielle en la cardinalité de S. En effet, pour un système implicatif minimal - i.e. sans règles - le treillis de fermés est composé des $2^{|S|}$ parties de S. A l'inverse, les bases canoniques d'un treillis minimal composé d'un ensemble S d'irréductibles incomparables entre eux et tous reliés à l'élément minimal et à l'élément maximal sont exponentielles en la cardinalité de S. Ainsi, les deux bases canoniques d'un treillis - la directe et la non directe - ont également une cardinalité dans le pire des cas exponentielle en la taille du treillis.

Ces problèmes de génération appartiennent à la classe des problèmes définis par une entrée de taille n, et un résultat de taille N bornée par 2^n. Une analyse classique de complexité au "pire des cas" ne peut qu'aboutir à des problèmes NP-difficiles en temps et en espace, alors même que la récente montée en puissance des ordinateurs rend possible le développement d'applications traitant de données ainsi générées.

C'est pourquoi une analyse dirigée à la fois par l'entrée et la sortie ([100]) apparaît plus pertinente de par l'estimation plus fine qu'elle propose. L'idée est de considérer le temps nécessaire à la génération d'un seul élément de la sortie. Deux techniques d'analyse sont proposées :

L'*analyse amortie* extrait le *coût amorti* - ou coût moyen - c de génération d'un élément à partir de la complexité classique $O(cN)$ où N est la taille de la sortie. On parlera ainsi de *complexité amortie polynomiale* lorsque c est borné par un polynôme.

L'*analyse incrémentale* considère quant à elle la liste ordonnée des éléments générés en sortie et cherche à calculer précisément le temps écoulé entre la génération de deux éléments consécutifs appelé *coût incrémental*. Un *algorithme incrémental polynomial* est alors un algorithme avec un coût incrémental borné par un polynôme.

En ce qui concerne la complexité en espace, la nécessité de stocker ou non la sortie est un indicateur pertinent qui permet de faire la distinction entre plusieurs types de problèmes de génération. Un *algorithme de dénombrement* consiste à compter le nombre d'éléments de la sortie ; un *algorithme de génération* génère explicitement le résultat, mais ne le stocke pas nécessairement ; un *algorithme de construction* génère et stocke le résultat.

Dans le domaine de la théorie des treillis, la majorité des problèmes de génération abordés sont des problèmes difficiles à sortie exponentielle :

Génération des irréductibles :

- `Entrée: treillis, contexte ou système de fermeture quelconque.` Problème polynomial.

Génération du treillis de fermés :

- `Entrée: système de fermeture quelconque.` Problème incrémental polynomial.

Génération de la base canonique :

- `Entrée: treillis.` Problème polynomial.
- `Entrée: système implicatif.` Problème polynomial.
- `Entrée: système de fermeture quelconque.` Problème ouvert. Bien que là, la NP-difficulté n'ait pas été démontrée, il n'existe pas à l'heure actuelle d'algorithme polynomial de génération d'une implication.

Génération de la base canonique directe : Problème équivalent à la génération du graphe de dépendance ou encore des générateurs minimaux.

- `Entrée: treillis.` Problème polynomial amorti.
- `Entrée: système de fermeture ou système implicatif.` Problème ouvert. Bien que la la NP-difficulté n'ait pas été démontrée, il n'existe pas à l'heure actuelle d'algorithme polynomial de génération d'une implication.

4.3 Génération des éléments irréductibles

La génération des éléments irréductibles est un problème qui diffère selon l'entrée considérée à savoir un treillis, un contexte, ou de façon générique un système de fermeture.

Les éléments irréductibles d'un treillis en entrée s'organisent classiquement sous forme de sa table, et leur génération revient à calculer la table du treillis. Lorsque l'entrée est un contexte, les éléments irréductibles correspondent à certains objets et attributs du contexte : un attribut (resp. un objet) est dit sup-irréductible (resp. inf-irréductible) si sa fermeture selon $\beta \circ \alpha$ (resp. $\alpha \circ \beta$) est un sup-irréductible (resp. inf-irréductible) qui ne contient pas d'autres attributs (resp. objets) avec la même fermeture. Un contexte qui ne contient que des objets et attributs irréductibles est un contexte réduit, et la génération des éléments irréductibles d'un contexte revient à en calculer la réduction. Les éléments irréductibles d'un contexte se caractérisent par ses opérateurs de fermeture. C'est pourquoi leur génération s'envisage de façon générique pour un système de fermeture quelconque, défini sur un ensemble S qu'il s'agit de réduire à ses seuls éléments irréductibles.

Pour un contexte comme pour un système de fermeture, c'est grâce aux propriétés d'un opérateur de fermeture que la génération des irréductibles est rendue possible, sans avoir à calculer l'ensemble des fermés du treillis dont le nombre est exponentiel dans le

pire des cas. C'est pourquoi le problème de génération des irréductibles est un problème polynomial à sortie polynomiale, quelque soit l'entrée considérée.

Génération des irréductibles : problème polynomial.
- `Entrée: treillis.`
- `Sortie: table du treillis.`

Un treillis $(S, \leq)$ est un graphe particulier dont le codage classique par liste de successeurs semble le plus approprié. Alors que la définition des irréductibles comme les éléments qui ne sont borne supérieure ou inférieure d'aucune partie ne les contenant pas (voir la définition 3) induit une génération d'un nombre exponentiel de parties, leur caractérisation par l'existence d'un unique prédécesseur ou successeur immédiat selon $\prec$ permet de les calculer en temps linéaire à partir du diagramme de Hasse. Les irréductibles d'un treillis s'organisent ensuite classiquement sous forme d'une table binaire. Par conséquent, le problème de génération des irréductibles d'un treillis se ramène à un calcul de sa réduction transitive, suivi du calcul de sa table.

L'algorithme de Goralcikova-Koubeck [94] est un standard de la littérature pour calculer à la fois la fermeture et la réduction transitive d'un graphe orienté sans cycle et en particulier d'un ordre ou d'un treillis. Cet algorithme exploite l'existence d'un tri topologique dans un graphe sans cycle, évitant ainsi de traiter récursivement tous les triplets de S^3 afin d'établir ou de supprimer la transitivité. La complexité en est par conséquent améliorée :

Complexité 1 *L'algorithme de Goralcikova-Koubeck [94] (cf Algorithme 3) calcule la réduction transitive d'un ordre* $(S, \leq)$ *en* $O(|S||\leq| + |S||\prec|)$.

La génération de la table binaire d'un treillis calcule tout d'abord l'ensemble J_L de ses sup-irréductibles et l'ensemble M_L de ses inf-irréductibles. Il s'agit ensuite de déterminer, pour chaque paire $(j, m) \in J_L \times M_L$, si $j \leq m$ ou $j \not\leq m$ par un simple parcours du treillis. Le calcul des relations flèches qui composent la table complète est un traitement similaire des irréductibles qui ne sont pas en relation selon $\leq$ et qui fait intervenir l'unique prédécesseur immédiat de chaque sup-irréductible, ainsi que l'unique successeur immédiat de chaque inf-irréductible. D'où une complexité polynomiale :

Complexité 2 *L'algorithme 4 calcule la table (binaire) d'un treillis* $(S, \leq)$ *en* $O(|J_L| \times |M_L| \times |S|)$.

Génération des irréductibles : Problème polynomial simple.
- `Entrée: contexte.`
- `Sortie: contexte réduit aux sup et inf-irréductibles.`

La génération des irréductibles d'un contexte consiste à en calculer sa réduction. L'algorithme de réduction décrit dans [82] réduit ainsi un contexte en $O(|O|^2|I| + |I|^2|O|))$.

Nom : `réductionTransitive`
Données : Un graphe sans cycle $G = (S, \leq)$.
Résultat : La réduction transitive $(S, \prec)$ de G.
début
  foreach $x \in S$ **do** marquer x;
  soit T un tri topologique de G;
  foreach $x \in S$ **do**
    soit *tmp* une copie de $succ(x)$;
    while S *non vide* **do**
      soit y le plus petit élément de *tmp* selon T;
      if y *non marqué* **then** ajouter l'arc (x, y) dans $\prec$;
      foreach $z \in succ(y)$ *tel que* z *n'est pas marqué* **do**
        ajouter l'arc (x, z) dans $\prec$; marquer z ; ajouter z dans *tmp*
      end
      supprimer y de *tmp*
    end
    foreach $y \in succ(x)$ **do** marquer y à faux
  end
fin

Algorithme 3: Réduction transitive d'un graphe sans cycles [94]

La réduction doit s'appliquer à la fois sur les objets - via l'opérateur de fermeture $\alpha \circ \beta$ - et sur les attributs - via l'opérateur de fermeture $\beta \circ \alpha$. De façon plus générique, il s'agit de générer les sup-irréductibles d'un système de fermeture quelconque.

Génération des irréductibles : Problème polynomial simple.
- `Entrée: système de fermeture.`
- `Sortie: système de fermeture réduit aux sup-irréductibles.`

Les éléments irréductibles d'un système de fermeture quelconque sont les éléments dont la fermeture est un sup-irréductible dans le treillis de fermés (voir la définition 10). Un système de fermeture (S, φ) est dit réduit lorsque l'ensemble S sur lequel il est défini ne contient que des sup-irréductibles. La génération des irréductibles revient donc à réduire le système de fermeture.

Nous proposons un algorithme générique de génération des irréductibles d'un système de fermeture. Cet algorithme s'applique aussi bien à un système de fermeture décrivant un contexte, qu'à un système de fermeture décrivant un système implicatif, avec la complexité suivante :

Complexité 3 *L'algorithme 5 réduit un système de fermeture quelconque* (S, φ) *en* $O(c_\varphi|S| + |S|^2 \log S)$.

Nom : `tableTreillis`
Données : Un treillis $L = (S, \leq)$ et son diagramme de Hasse $\prec$
Résultat : La table T du treillis L
début
soit J l'ensemble des sup-irréductibles (un seul arc entrant) ;
soit M l'ensemble des inf-irréductibles (un seul arc sortant) ;
initialiser une table T avec J en lignes et M en colonnes;
foreach $(j, m) \in J \times M$ **do**
si $j \leq m$ **alors** $T[m, j] = \times$;
else
soit j^- l'unique prédécesseur de j selon $\prec$;
soit m^+ l'unique successeur de m selon $\prec$;
si $j \leq m^+$ *et* $j^- \leq m$ **alors** $T[m, j] = \updownarrow$;
si $j \leq m^+$ *et* $j^- \not\leq m$ **alors** $T[m, j] = \uparrow$;
si $j \not\leq m^+$ *et* $j^- \leq m$ **alors** $T[m, j] = \downarrow$;
si $j \not\leq m^+$ *et* $j^- \not\leq m$ **alors** $T[m, j] = \circ$;
end
end
retourner T
fin

Algorithme 4: Génération de la table d'un treillis

Rappelons qu'un système de fermeture non réduit contient des éléments réductibles, un élément $x \in S$ étant réductible lorsque $\varphi(x)$ n'est pas un sup-irréductible, i.e. lorsqu'il est la borne supérieure d'un ensemble d'éléments ne le contenant pas (voir l'équation 2.21 de la section 2.3). La réduction consiste alors à déterminer, pour chaque élément $x \in S$ s'il existe un ensemble $E_x \subseteq S$ d'éléments équivalents tel que $x \notin E_x$ et $\varphi(x) = \varphi(E_x) = \vee E_x$. Lorsque c'est le cas, x est alors réductible, et peut être remplacé par E_x dans le système de fermeture.

Alors qu'une application directe de la définition de E_x induit un coût exponentiel de détection d'un réductible, l'utilisation de la relation de précédence *permet une réduction polynomiale (cf algorithme 5). Elle met en relation deux éléments x et y de S lorsque $\varphi(x) \subseteq \varphi(y)$. Elle correspond ainsi au sous-ordre induit par ses sup-irréductibles lorsque le système de fermeture est réduit, alors qu'elle peut contenir des cycles si le système de fermeture n'est pas réduit. Nous proposons un algorithme générique de réduction en deux étapes :*

Etape 1. *Il s'agit tout d'abord de remplacer chaque composante fortement connexe $X \subseteq S$ du graphe de précédence par un unique élément représentatif $y \in X$. En effet, les éléments d'une même composante connexe sont des éléments de même fermeture qui, clairement, ne vérifient par l'équation 2.21, et sont donc réductibles. Cependant, une fois que tous les éléments de X sauf y sont supprimés du système de fermeture, alors l'élément restant y doit être conservé car l'ensemble $X \setminus y$ dont il était équivalent a été supprimé. On peut observer que le graphe de précédence du système de fermeture ainsi modifié est alors acyclique, possédant la propriété d'un ordre.*

Etape 2. *Une fois les composantes fortement connexes traitées, il s'agit de tester, pour chaque élément $x \in S$, si $\varphi(x) = \varphi(E_x) = \vee E_x$, avec E_x prédécesseurs immédiats de x dans le graphe de précédence. En cas d'égalité, x est réductible, et la réduction consiste alors à le remplacer par les éléments de E_x qui lui sont équivalents. On peut remarquer qu'un élément x possédant un seul prédécesseur immédiat y ne peut être réductible, car x et y seraient alors équivalents, appartenant ainsi à une même composante connexe déjà traitée à l'étape 1. Les seuls éléments pouvant être réductibles sont donc des éléments possédant soit aucun, soit plusieurs prédécesseurs immédiats. Un élément x sans prédécesseurs immédiats est réductible si $\varphi(x) = \varphi(\emptyset) = \bot$, la réduction consistant alors à le supprimer.*

Name : `réductionSystèmeFermeture`
Data : un système de fermeture (S, φ)
Result : l'ensemble $X \subset S$ des éléments réductibles, et l'ensemble E_x des éléments équivalents pour chaque $x \in X$
début
  $Res = \emptyset$;
  Initialiser un graphe G avec S comme ensemble des noeuds;
  foreach $(x, y) \in S \times S$ **do**
    if $\varphi(x) \subseteq \varphi(y)$ **then** ajouter l'arc (x, y) dans G;
  Calculer l'ensemble CFC des composantes fortement connexes de G;
  foreach $C \in CFC$ **do**
    Choisir un représentant $y \in C$;
    foreach $x \in C$ *tel que* $x \neq y$ **do**
      ajouter x dans Res avec $E_x = \{y\}$;
      supprimer x du graphe G;
  foreach $x \in G$ **do**
    Soit P l'ensemble des prédécesseurs immédiats de x dans le graphe G;
    if $|P| \neq 1$ *et* $\varphi(x) = \varphi(P)$ **then**
      ajouter x dans Res avec $E_x = P$
  retourner Res;
fin

Algorithme 5: Réduction d'un système de fermeture à ses sup-irréductibles

4.4 Génération du treillis de fermés

Les algorithmes de génération du treillis des concepts d'un contexte, ou encore du treillis des fermés d'un système implicatif, reposent sur les propriétés de l'opérateur de fermeture. C'est pourquoi nous abordons ce problème sous sa forme générique, à savoir la génération du treillis des fermés d'un système de fermeture quelconque (S, φ). Il s'agit d'un problème de génération à sortie exponentielle, la taille du treillis pouvant être exponentielle.

Un grand nombre d'algorithmes de génération d'un treillis ont été proposés dans la littérature avec différentes stratégies. Le premier algorithme, qui date de 1969, [87], calcule tous les concepts d'un contexte par extraction de ses sous-matrices premières. L'algorithme NextClosure de Ganter, algorithme de référence proposé en 1984 [91], calcule l'ensemble des fermés selon un ordre lectique, alors que celui de Bordat [86] est le premier algorithme qui génère directement le diagramme de Hasse du treillis de fermés. On trouve ensuite un grand nombre d'algorithmes dans la littérature, certains incré-

mentaux [105, 92]. Dans de récents travaux, Gély [93] propose un algorithme générique dans un même contexte unifié pour les problèmes de génération de treillis.

Tous ces algorithmes génèrent un élément du treillis en temps incrémental polynomial. La meilleure complexité théorique est due à Nourine and Raynaud in [107] avec une complexité en espace exponentielle car l'ensemble des fermés est conservé dans uns structure d'arbre. Dans une étude comparative à la fois théorique et expérimentale [98] il apparaît que les temps de calcul varient beaucoup à la fois selon les données et les algorithmes. Une étude comparative sur l'impact de la structure de données utilisée - tableau de bits, liste chaînée triée, arbre binaire de recherche, table de hachage - aboutit au même résultat [97].

Nous décrivons ici l'algorithme NextClosure [91] (cf algorithme 6), et l'algorithme de Bordat [86] (cf algorithme 7), algorithmes fondateurs de génération d'un treillis de fermés qui illustrent les deux principales stratégies de génération : génération de la famille de fermés, ou bien du diagramme de Hasse du treillis des fermés.

Génération du treillis de fermés : Problème incrémental polynomial

- `Entrée: système de fermeture.`
- `Sortie: famille - ou diagramme de Hasse du treillis - des fermés du système de fermeture.`

L'algorithme NextClosure proposé par Ganter [91] (cf Algorithme 6) calcule le suivant d'un fermé selon l'*ordre lectique* défini sur les fermés. La génération de l'ensemble des fermés consiste alors à calculer le suivant de chaque fermé, à partir du fermé $\varphi(\emptyset)$. Le treillis, ou son diagramme de Hasse, s'obtient ensuite en ordonnant les concepts par inclusion.

L'ordre lectique se définit pour un ensemble $S = x_1, \ldots x_n$ d'éléments indicés. Un fermé A est inférieur lectiquement à un fermé B si l'élément de plus petit indice i qui distingue A et B appartient à B, on note alors $A <_i B$:

$$A <_i B \text{ ssi } \exists x_i \in B \setminus A \text{ tel que } A|i = B|i \tag{4.1}$$

avec $A|i = A \cap \{x_1, x_2, \ldots x_{i-1}\}$, restriction de A aux $(i-1)$ premiers éléments de S. Par exemple, pour $S = \{a, b, c\}$, on aura $ac <_1 bc$, $ab <_2 ac$ et $ab <_3 abc$. Clairement, l'ordre lectique définit un ordre total sur les parties de S et étend l'ordre d'inclusion - i.e. si $A \subseteq B$ alors il existe $i \leq n$ tel que $A \leq_i B$.

La complexité en temps de l'algorithme NextClosure s'exprime en fonction du coût c_φ de génération d'une fermeture, coût qui dépend avant tout du système de fermeture :

Complexité 4 *[91] L'algorithme NextClosure (algorithme 6) calcule l'ensemble des fermés d'un système de fermeture en $O(|S|c_\varphi)$ par fermé, avec une complexité polynomiale en espace - tous les fermés n'ont pas à être stockés.*

Pour un contexte $(O, I, (\alpha, \beta))$ la complexité de NextClosure est en $O(|O|^2|I|)$ ou en $O(|I|^2|O|)$ par fermé selon que l'on considère le système de fermeture défini sur

l'ensemble des attributs I, ou l'ensemble des objets O. Pour un système d'implication Σ quelconque, elle est en $O(|S|^2|\Sigma|)$ par fermé.

Nom : `nextClosure`
Données : Un système de fermeture (S, φ) ; un fermé F
Résultat : Le suivant de F selon l'ordre lectique
début
  $K = S \setminus F$ trié par ordre décroissant des indices des éléments de S;
  foreach $x_i \in K$ **do**
    $A = \varphi(F|i + x_i)$;
    si $F <_i A$ **alors** retourner A
  end
fin

Algorithme 6: Génération du suivant d'un fermé selon l'ordre lectique

L'algorithme de Bordat [86] a pour principale caractéristique de calculer directement le diagramme de Hasse du treillis. Introduit dans le cas particulier où le système de fermeture est un contexte $(O, I, (\alpha, \beta))$, cet algorithme génère récursivement à partir du concept minimal les successeurs immédiats d'un concept dans le diagramme de Hasse. L'algorithme de Bordat s'étend facilement à un système de fermeture quelconque, les sucesseurs immédiats d'un fermé sont alors générés récursivement à partir du fermé minimal $\varphi(\emptyset)$ (cf Algorithme 7).

Il est ici important de noter qu'un fermé peut être généré plusieurs fois. En effet, un test d'existence de chaque fermé généré est nécessaire avant de relancer récursivement le calcul de ses successeurs immédiats. A l'inverse, l'algorithme NextClosure ne génère qu'une seule fois chaque fermé. En revanche, la structure du diagramme de Hasse du treillis est obtenue directement par l'algorithme de Bordat, ce qui n'est pas le cas avec l'algorithme NextClosure.

Dans sa forme initiale, l'algorithme de Bordat calcule les successeurs immédiats d'un concept (A, B) sur la base du théorème de Bordat [86] qui établit qu'ils sont en bijection avec les sous-ensembles maximaux par inclusion de la famille $\mathcal{F}_A$ définie sur l'ensemble des objets par :

$$\mathcal{F}_A = \{\beta(x) \cap A \ : \ x \in I \setminus B\} \tag{4.2}$$

Pour un système de fermeture quelconque, il s'agit de calculer les successeurs immédiats d'un fermé. Une extension du théorème de Bordat, que nous avons décrite dans [8], établit que les successeurs immédiats d'un fermé sont en bijection avec les sous-ensembles minimaux par inclusion de la famille $\mathcal{F}_F$ définie sur $S \setminus F$ par :

$$\mathcal{F}_F = \{\varphi(F + x) \ : \ x \in S \setminus F\} \tag{4.3}$$

Les successeurs immédiats d'un fermé F sont ainsi calculés en deux temps : la famille $\mathcal{F}_F$ est tout d'abord générée en $O(|S|c_\varphi)$, avec c_φ coût de génération d'une

fermeture puis ses sous-ensembles minimaux par inclusion sont générés en $O(|S|^3)$. L'algorithme 8 en présente une implémentation qui utilise le graphe d'inclusion des parties de $\mathcal{F}_F$. L'utilisation de structures de données sophistiquées permet de résoudre le problème des sous-ensembles minimaux par inclusion en $O(|S|^{2,5})$. La complexité en espace est quant à elle exponentielle car l'ensemble des fermés doit être stocké.

Complexité 5 *[86] L'extension de l'algorithme de Bordat (Algorithme 7) calcule l'ensemble des fermés d'un système de fermeture en $O(|S|c_\varphi + |S|^{2,5})$ par fermé, avec une complexité exponentielle en espace.*

Lorsque le système de fermeture est un contexte, on retrouve la complexité en $O(|I|^2|O|)$ par fermé de l'algorithme de Bordat. Pour mieux comprendre l'extension du théorème de Bordat, considérons le système de fermeture défini sur l'ensemble des attributs I d'un contexte $(O, I, (\alpha, \beta))$ par l'opérateur de fermeture $\varphi = \beta \circ \alpha$. Dans le treillis de fermés associé, chaque fermé correspond à la partie B d'un concept (A, B). Pour deux attributs x et x' de B, on a alors l'équivalence suivante sur laquelle repose l'extension du théorème de Bordat :

$$\beta(x) \cap A \subseteq \beta(x') \cap A \iff \varphi(x + B) \supseteq \varphi(x' + B) \tag{4.4}$$

En effet, si $\beta(x) \cap A = \beta(x') \cap A$, alors x et x' appartiennent au même concept successeur de (A, B). Ce qui signifie que $\varphi(x+B) = \varphi(x'+B)$. Dans le cas contraire, seul le plus grand par inclusion des deux ensembles $\beta(x) \cap A$ et $\beta(x') \cap A$ est susceptible d'être successeur immédiat de (A, B). Ce qui revient à considérer le plus petit par inclusion des deux ensembles $\varphi(x + B)$ et $\varphi(x' + B)$.

Les propriétés des treillis manipulés sont importantes pour identifier l'algorithmique adaptée. En effet, pour certains types de treillis il existe des algorithmes plus efficaces que dans le cas général. C'est en particulier le cas des treillis distributifs, pour lesquels l'algorithme de génération le plus efficace [95] exploite les propriétés de duplication : le treillis distributif est généré par duplications successives d'intervalles particuliers, ce qui améliore grandement la complexité.

La relation de couverture générée par l'algorithme de Bordat offre l'avantage de permettre un mécanisme de génération à la demande d'un treillis pour ne générer que la portion du treillis nécessaire à une tâche donnée. Une telle extension est pertinente lorsque le treillis est utilisé comme un espace de recherche qu'il s'agit de parcourir selon son diagramme de Hasse. Nous avons proposé une méthode de classification par navigation dans un treillis qui propose ce mécanisme de génération à la demande, d'où un gain considérable en temps de calcul. Le treillis est ainsi parcouru à la demande par navigation à partir de l'élément minimal, et ce jusqu'à ce qu'un inf-irréductible soit atteint. Cette méthode, utilisée dans un contexte de reconnaissance de symboles, est décrite dans la section 6.3.1 de la partie I.

De façon plus générale, on trouve dans [55] un mécanisme de recherche d'information à la fois par requête et par navigation : à chaque requête de l'utilisateur correspond un

Name : `Bordat`
Données : Un système de fermeture $C = (S, \varphi)$
Résultat : Le diagramme de Hasse $(\mathcal{F}, \prec)$ du treillis de fermés de C
début
  $\mathcal{F} = \{\varphi(\emptyset)\}$;
  foreach $F \in \mathcal{F}$ *non encore marqué* **do**
    succ=`successeursImmédiats` (C, F);
    foreach $X \in$ *succ* **do**
      $F' = F + X$;
      if $F' \notin \mathcal{F}$ **then** ajouter F' dans $\mathcal{F}$;
      ajouter l'arc $F \prec F'$
    end
    marquer F
  end
  retourner $(\mathcal{F}, \prec)$
fin

Algorithme 7: Génération du diagramme de Hasse du treillis de fermés d'un système de fermeture

Nom : `successeursImmédiats`
Data : Un système de fermeture (S, φ) ; un fermé F de son treillis de fermés
Result : Les successeurs de F dans le diagramme de Hasse du treillis de fermés
début
  initialiser la famille $\mathcal{F}_F$ à vide;
  foreach $x \in S \setminus F$ **do**
    ajouter $\varphi(x)$ dans $\mathcal{F}_F$
  end
  initialiser un graphe G avec pour noeuds les parties de $\mathcal{F}_F$;
  foreach *parties distinctes* X *et* Y *de* $\mathcal{F}_F$ **do**
    if $X \subseteq Y$ **then** ajouter dans G l'arc (X, Y)
  end
  retourner l'ensemble des éléments minimaux de G';
fin

Algorithme 8: Calcul des successeurs immédiats d'un fermé

fermé dont les successeurs et prédécesseurs lui sont soumis pour qu'il puisse affiner son choix.

Enfin, nous présentons l'algorithme 10 qui est une version non récursive de l'algorithme de Bordat. Cette version non récursive en permet une exploitation pour générer un treillis à l'aide de plusieurs processeurs. Nous avons mené des expérimentations en ce sens, et obtenu de bons résultats (cf section 6.4.2 de la partie I), résultats qui s'expliquent en partie par l'utilisation d'un ensemble partagé par tous les processeurs pour stocker les arcs du treillis, et non les noeuds, au fur et à mesure de leur génération.

Nom : BordatItératif **Données** : Un système de fermeture $C = (S, \varphi)$
Résultat : Le diagramme de hasse du treillis de fermés de C
début
  initialiser l'ensemble S des noeuds à traiter avec $\varphi(\emptyset)$;
  initialiser l'ensemble R des noeuds calculés à vide;
  initialiser l'ensemble E des arcs du treillis à vide;
  foreach $n \in S$ **do**
    $succ =$ `successeursImmédiats`(n);
    foreach $n' \in succ$ **do**
      si $n' \notin R$ **alors** ajouter n' dans R et dans S;
    end
    supprimer n de S, l'ajouter dans R;
    ajouter l'arc (n, n') dans E;
  end
  retourner (S, E) ;
fin

Algorithme 10: Version itérative de la génération du diagramme de Hasse du treillis de fermés d'un système de fermeture

Nous avons proposé différentes extensions de l'algorithme de Bordat [8]. Nous avons également proposé dans [16] une génération incrémentale de la famille des fermés d'un système implicatif Σ sur la base de la propriété suivante :

$$\mathcal{F} = \{\varphi(\emptyset)\} \cup \{\varphi(x) \,|\, x \in S\} \cup \{\varphi(F_1 \cup F_2) \,|\, F_1, F_2 \in \mathcal{F}\}$$

4.5 Génération de la base canonique

Bien que la base canonique soit de taille inférieure à la base canonique directe, elle peut être exponentielle en la taille de l'ensemble S qui définit un système de fermeture. C'est pourquoi sa génération appartient alors à la classe des problèmes à sortie exponentielle. Cependant, lorsque l'entrée est un treillis, ou un système implicatif, alors la base canonique est polynomiale en la taille de l'entrée, et s'obtient en temps polynomial.

Génération de la base canonique : Problème polynomial
- `Entrée: système implicatif.`
- `Sortie: base canonique du système implicatif.`

La taille de la base canonique Σ_{cb} d'un système implicatif Σ est par définition inférieure à celle de Σ. Les prémisses de ses règles sont définies comme les pseudo-fermés du treillis de fermés. Sa génération découle de la propriété suivante qui caractérise un pseudo-fermé P comme étant égal à sa fermeture après suppression de la règle $P \rightarrow \varphi(P) \backslash P$ [115] :

$$P = \varphi_{\Sigma_{cd} \backslash \{P \rightarrow \varphi(P) \backslash P\}}(P) \tag{4.5}$$

Cette caractérisation induit une génération de Σ_{cd} en trois étapes :

1. Transformer Σ en un SI maximal à droite : il s'agit de remplacer la conclusion de chaque règle $X \rightarrow Y$ de Σ par $\varphi(X) \setminus X$.
2. Transformer Σ en un SI minimum : il s'agit de vérifier si la suppression de chaque règle $X \rightarrow Y$ de Σ modifie ou non la fermeture de X. Lorsque c'est le cas, la règle peut être supprimée.
3. Générer les pseudo-fermés en remplaçant la prémisse de chaque règle $X \rightarrow Y$ de Σ par la fermeture de X dans $\Sigma' = \Sigma \backslash \{X \rightarrow Y\}$.

Complexité 6 *L'algorithme 11 calcule la base canonique d'un SI Σ défini sur S en $O(|\Sigma|^2|S|^3)$.*

Nom : `baseCanonique`
Données : un SI Σ
Résultat : la base canonique équivalente à Σ
début
  //transformation maximale à droite;
  foreach $X \rightarrow Y \in \Sigma$ **do**
    remplacer $X \rightarrow Y$ par $X \rightarrow \varphi(X) \setminus X$
  //transformation minimum;
  foreach $X \rightarrow Y \in \Sigma$ **do**
    $\Sigma' = \Sigma \backslash \{X \rightarrow Y\}$;
    if $\varphi_{\Sigma'}(X) = \varphi_{\Sigma}(X)$ **then**
      supprimer $X \rightarrow Y$ de Σ
  //génération pseudo-fermés;
  foreach *règle* $X \rightarrow Y \in \Sigma$ **do**
    $\Sigma' = \Sigma \setminus \{X \rightarrow Y\}$;
    remplacer $X \rightarrow Y$ par $\varphi_{\Sigma'}(X) \rightarrow Y$
  retourner Σ
fin

Algorithme 11: Génèration de la base canonique d'un SI

Génération de la base canonique : Problème polynomial

- `Entrée: treillis.`
- `Sortie: base canonique du treillis.`

La base canonique d'un treillis est de taille polynomiale en celle du treillis. L'algorithme de Shock [112], introduit dans le domaine des bases de données, en propose une génération en deux étapes :

1. Produire le SI Σ_L pour un treillis $L = (X, \leq)$, polynomial en la taille du treillis qui se définit à partir du treillis de fermés $(J_x, \subseteq)_{x \in X}$ isomorphe à L par :

$$\begin{aligned} \Sigma_L \quad = \quad & \{A \rightarrow J_y \setminus A \, : \, A \in succ(x) \text{ pour } x \in X \text{ et} && (4.6) \\ & \exists y \in X \text{ tq } A \subseteq J_y \text{ et } J_y \text{ minimal pour cette propriété}\} && (4.7) \end{aligned}$$

   avec, pour $x \in X$:

$$succ(x) = \cup\{J_x + j \, : \, j \in J_L \text{ et } \forall y \in X \, , J_x + j \neq J_y\} \quad (4.8)$$

2. Appliquer l'algorithme `baseCanonique` pour générer la base canonique de Σ.

Le SI Σ_L ainsi défini est représentatif du treillis [112]. Par ailleurs, il est de taille polynomiale en celle du treillis, bornée par $|X| * |J_L|$.

Complexité 7 *[112] L'algorithme de Shock (algorithme 12) calcule la base canonique d'un treillis $L = (X, \leq)$ en $O(|X|^2 * |J_L|)^3)$.*

Nom : `baseCanonique`
Données : un treillis $L = (X, \leq)$
Résultat : la base canonique du treillis
début
  //production d'un SI Σ_L pour le treillis;
  calculer l'ensemble J_L des sup-irréductibles ;
  initialiser la famille $\mathcal{F}$ à vide;
  initialiser le SI Σ_L à vide;
  foreach $x \in X$ **do**
    calculer J_x;
    ajouter J_x dans $\mathcal{F}$;
  foreach $x \in X$ **do**
    foreach $j \in J_L$ **do**
      soit $P = J_x + j$;
      if $P \notin \mathcal{F}$ **then**
        soit $F \in \mathcal{F}$ minimal tel que $P \subset F$;
        ajouter $P \rightarrow F \backslash P$ dans Σ_L
  // retourner la base canonique de Σ_L;
  retourner `baseCanonique`(Σ_L)
fin

Algorithme 12: Génèration de la base canonique d'un treillis

Génération de la base canonique : Problème ouvert
- `Entrée: système de fermeture.`
- `Sortie: base canonique du système de fermeture.`

La taille de la base canonique d'un système de fermeture quelconque peut-être exponentielle [99]. Le problème de sa génération est un problème ouvert, ce qui signifie qu'il n'existe pas à l'heure actuelle d'algorithme de génération polynomiale d'une implication à partir d'un système de fermeture quelconque, et en particulier à partir d'un contexte. Cependant, il a récemment été montré des résultats de co-NP-complétude sur la reconnaissance des pseudos-fermés [85], ce qui est mauvais signe pour espérer une génération polynomiale par implication.

La stratégie usuelle de génération de la base canonique d'un système de fermeture passe par une génération préalable du treillis de fermés à partir duquel la base canonique se génère en temps polynomial en la taille du treillis.

Dans [108] est proposé un algorithme de construction incrémentale de la base canonique à partir d'un contexte par ajout incrémental d'attributs.

4.6 Génération de la base canonique directe

La génération de la base canonique directe est un problème qui diffère selon l'entrée considérée, cette génération pouvant s'envisager à partir du treillis lui-même, mais également à partir d'un système de fermeture quelconque. Bien que les algorithmes les plus utilisés pour générer la base canonique directe à partir d'un treillis sont de complexité exponentielle amortie, il existe cependant des algorithmes polynomiaux amortis. En revanche, il s'agit d'un problème ouvert lorsque l'entrée est un système de fermeture. Bien que la NP-difficulté n'ait pas été démontrée, il n'existe pas à l'heure actuelle d'algorithme polynomial de génération d'une implication. C'est pourquoi la stratégie la plus commune consiste à passer par la génération du treillis de fermés avant d'en extraire sa base canonique directe, d'où un coût exponentiel. Il existe des algorithmes spécifiques dans le cas particulier où le système de fermeture est associé à un système implicatif, avec toujours une complexité exponentielle amortie.

Les différentes propriétés de la base canonique directe que nous avons identifiées dans la section 3.2.2, et en particulier celles issues de l'équivalence entre cinq bases décrites sous des formes différentes, font apparaître que le problème de génération de la base canonique directe est équivalent à la génération du graphe de dépendance (voir Définition 7) et des générateurs minimaux des éléments du treillis de fermés. En effet, la base canonique directe est composée des règles d'implication $X + j' \rightarrow j$ définies sur l'ensemble des sup-irréductibles J_L du treillis, où (j, j') est un arc du graphe de dépendance valué par $X \subseteq J_L$. Les prémisses des règles sont les générateurs minimaux du treillis.

Génération de la base canonique directe : Problème polynomial amorti.

- `Entrée: treillis.`
- `Sortie: base canonique directe du treillis (ou son graphe de dépendance).`

Le nombre de générateurs minimaux du treillis est exponentiel dans le pire des cas en la taille du treillis. Par conséquent, il en est de même pour la taille de la base canonique directe et du graphe de dépendance. Une analyse dirigée par la sortie est donc plus appropriée, qui considère la complexité de génération d'une Σ_{cdb}-implication, d'un générateur minimal ou encore d'une valuation du graphe de dépendance. Alors que les algorithmes les plus utilisés sont des algorithmes incrémentaux exponentiels, il existe cependant des algorithmes incrémentaux polynomiaux.

La définition des générateurs minimaux d'un élément d'un treillis (voir la définition 4) en induit une génération exponentielle. On retrouve la même complexité en exploitant l'équivalence entre les générateurs minimaux et les transversales minimales de chaque fermé du treillis (cf Section 3.2.2) problème connu pour être exponentiel. Citons en particulier l'algorithme incrémental de Pfaltz [109] ou encore l'algorithme Jen [88] qui calculent les générateurs minimaux d'un treillis.

Cependant, en logique, Ibaraki et al. ont proposé un algorithme [63] qui calcule les dépendances fonctionelles minimales - i.e. la base canonique directe - en temps polynomial amorti avec une famille $\mathcal{F}$ de fermés en entrée.

L'algorithme 13 que nous proposons repose sur la même stratégie pour générer le graphe de dépendance d'un treillis (voir la définition 7) en temps polynomial amorti. Rappelons que les noeuds du graphe sont les sup-irréductibles du treillis, et qu'il existe un arc entre deux sup-irréductibles j et j' lorsqu'ils sont en relation selon la relation de dépendance δ du treillis. La relation δ se calcule en temps polynomial par un simple calcul de bornes supérieures : il s'agit de déterminer, pour chaque $x \in S$ et pour chaque paire $(j, j') \in J \times J$, si j est en relation avec j' selon δ_x, i.e. si $x \vee j' \leq j$. La difficulté vient ensuite du calcul des générateurs minimaux de x qui valuent l'arc $j\delta j'$. L'algorithme 13 exploite la définition récursive des générateurs minimaux pour obtenir une complexité incrémentale polynomiale en traitant les éléments du treillis selon un tri topologique.

En effet, si on considère deux sup-irréductibles tels que $j\delta_x j'$ - avec x élément du treillis - ou de façon équivalente $j\delta_B j'$ - avec B générateur minimal de x - alors $B + j'$ est un générateur minimal de $x \vee j'$, défini récursivement à partir de B. On distingue deux cas :

- *Si B est une partie stricte de J_x, alors B se définit récursivement de la même façon à partir d'un générateur minimal d'un prédécesseur de x. Par conséquent, on le retrouve nécessairement dans le sous-graphe du graphe de dépendance induit par J_x, à la fois par les noeuds et par les valuations d'arcs.*
- *Dans le cas contraire, alors $B = J_x$ est l'unique générateur minimal de x.*

Complexité 8 *L'algorithme 13 construit le graphe de dépendance d'un treillis $L = (S, \leq)$ avec une complexité en $O(|\Sigma_{cdb}||S||J_L|^3)$, soit un coût amorti polynomial en $O(|S||J_L|^3)$ par Σ_{cdb}-implication ou par générateur minimal.*

Dans [8], nous proposons une extension de l'algorithme de Bordat pour une génération du graphe de dépendance pendant la construction du treillis. Le théorème de Bordat, sur lequel repose l'algorithme de génération du diagramme de Hasse d'un système de fermeture (S, φ), utilise implicitement la relation de dépendance δ du treillis de fermés, introduite dans la section 2.2.2. En effet, on montre facilement que la relation de dépendance se définit par $x\delta_F y$ si $\varphi(F + x) \subseteq \varphi(F + y)$, avec $x, y \in S \setminus F$ pour tout fermé F. Ainsi, elle peut se calculer linéairement pour chaque fermé F calculé. Il est également envisageable de calculer simultanément les générateurs minimaux de F pour en valuer les arcs du graphe de dépendance. Pour cela, il suffit de déterminer à la fois si F est un fermé minimal dans le treillis mettant en relation $x\delta y$, puis de retrouver ses générateurs minimaux par parcours du graphe de dépendance déjà calculé selon la stratégie décrite dans l'algorithme 13.

Génération de la base canonique directe : problème ouvert.

Nom : `grapheDépendance`
Données : Un treillis $L = (S, \leq)$
Résultat : Le graphe de dépendance du treillis
début
  calculer l'ensemble J des sup-irréductibles;
  initialiser un graphe G avec pour noeuds les éléments de J;
  foreach $x \in S$ **do**
    foreach $(j, j') \in J \times J$ *tel que* $x \vee j' \geq j$ **do**
      ajouter l'arc (j, j') dans G;
      calculer J_x ; initialiser la famille GM_x à vide;
      soit G' sous-graphe de G induit par J_x pour ses noeuds et ses valuations;
      foreach *arc* (k, k') *de* G' **do**
        foreach *valuation* B *de l'arc* (k, k') **do**
          si $B \vee k' = x$ **alors** ajouter l'ensemble $B + k'$ à la famille GM_x
        end
      end
      si GM_x *est vide* **alors** $GM_x = \{J_x\}$;
    end
  end
  valuer l'arc (j, j') avec GM_x;
  retourner G;
fin

Algorithme 13: Génération du graphe de dépendance d'un treillis

- `Entrée: système de fermeture.`
- `Sortie: base canonique directe du système de fermeture.`

L'approche classique de génération de la base canonique directe à partir d'un système de fermeture consiste à passer par la génération du treillis de fermés avant d'en extraire sa base canonique directe, d'où un coût exponentiel qui intègre la génération des fermés. Il est ainsi possible d'en déduire les générateurs minimaux du treillis, ou encore son graphe de dépendance.

Citons, en fouille de données, l'algorithme de Taouil and Bastide dans [80] qui utilise la stratégie de calcul des transversales minimales de chaque fermé du treillis pour les implications minimales à gauche - i.e. les Σ_{cdb}-implications - à partir d'un contexte en temps et en espace exponentiel par implication, ou encore l'algorithme incrémental de Pfaltz [109] et l'algorithme jen [88] qui calculent les générateurs minimaux - i.e. les prémisses de la base canonique directe - d'un contexte.

Comme nous l'avons déjà souligné, l'utilisation de l'algorithme 13 que nous proposons qui nécessite un traitement du treillis de fermés selon un tri topologique, peut s'intégrer directement au calcul des successeurs immédiats sur la base d'une adaptation du théorème de Bordat [26] :

Complexité 9 *Les algorithmes 7 et 13 combinés permettent d'obtenir la base canonique directe d'un système de fermeture en $O(|\Sigma_{cdb}|^2|\mathcal{F}|^2|J_{\mathcal{F}}|^2)$, soit un coût amorti exponentiel en $O(|\mathcal{F}|^2|J_{\mathcal{F}}|^2)$ par Σ_{cdb}-implication, avec $\mathcal{F}$ treillis de fermés dont la taille est exponentielle en celle de l'ensemble S.*

Citons également l'algorithme de Mannila dans [103] qui accepte en entrée les éléments irréductibles du système de fermeture - i.e. sa réduction - et génère sa base canonique directe avec une complexité exponentielle par implication. Citons également l'algorithme avec la meilleure complexité connue proposé par Fredman and Khachiyan ([89]) qui prend une formule booléenne sous forme DNF - équivalent à un SI - et qui génère une implication en $O(|S|^{\log|S|})$, i.e en temps quasi-polynomial. Une modification en a récement été proposée dans [96] avec une première étape déterministe en temps amorti polynomial, suivie d'étapes non déterministes en $O(log^2|S|)$.

Génération de la base canonique directe : problème ouvert.

- `Entrée: système implicatif.`
- `Sortie: base canonique directe du système implicatif.`

Lorsque le système de fermeture est lui-même un système implicatif Σ, on trouve alors plusieurs algorithmes de transformation de Σ en sa base canonique directe. Cependant, la base canonique directe Σ_{cdb} peut être exponentielle dans le pire des cas en la taille du SI Σ, et tout algorithme de génération doit être analysé en considérant le coût de génération d'une implication de la base.

Dans [116], Wild propose un algorithme en temps exponentiel par implication, qui génère un SI intermédiaire et plus grand, de taille exponentielle dans le pire des cas. La

base directe-optimale que nous avons introduit dans [16] - une des cinq bases égales à la base canonique directe, voir section 3.2.2 - est définie à partir d'un SI quelconque Σ par l'application successive des traitements suivants (cf algorithme 14) dont le deuxième génère également un SI intermédiaire et exponentiel :

1. Transformer Σ en un SI unaire ;
2. Transformer Σ en un SI direct dont la taille peut être exponentielle en fonction de celle de Σ ;
3. Transformer Σ en un SI minimal à gauche en conservant, pour deux règles de même conclusion, celle dont la prémisse est minimale par inclusion ;
4. Transformer Σ en un SI compact en remplaçant par une seule règle toutes les règles de même prémisse.

Complexité 10 *L'algorithme 14 calcule la base canonique d'un SI Σ sur S en $O(2^{|S|})$.*

Nom : `baseCanoniqueDirecte`
Data : un SI Σ
Result : la base canonique directe équivalente à Σ
début
  //Transformer Σ en un SI unaire;
  //Transformer Σ en un SI direct;
  foreach *règle* $X \rightarrow y \in \Sigma$ **do**
    foreach *règle* $X' \rightarrow y' \in \Sigma$ **do**
      if $y \in X'$ *et* $y' \notin X$ **then**
        ajouter $X \cup X' \backslash \{y\} \rightarrow y'$ à Σ
  //Transformer Σ en un SI minimal à gauche;
  foreach *règle* $X \rightarrow y \in \Sigma$ **do**
    foreach *règle* $X' \rightarrow y' \in \Sigma$ **do**
      if $X \subseteq X'$ *and* $y = y'$ **then**
        supprimer $X' \rightarrow y'$ de Σ
  //Transformer Σ en un SI compact;
  foreach *règle* $X \rightarrow Y \in \Sigma$ **do**
    foreach *règle* $X' \rightarrow Y' \in \Sigma$ **do**
      if $X = X'$ **then**
        remplacer $X \rightarrow Y$ et $X \rightarrow Y'$ par $X \rightarrow YY'$ dans Σ
  retourner Σ
fin

Algorithme 14: Génération de la base canonique directe d'un SI

Le premier traitement qui définit la base direct-optimale est similaire au calcul des clauses résolvantes en logique, stratégie qui peut s'utiliser dans un mécanisme d'inférence par chaînage à partir de règles. Le lien avec les fonctions booléennes est ici intéressant. En effet, le nombre de clauses résolvantes est exponentiel en fonction de la taille de la formule (i.e. nombre de variables), sauf lorsque les clauses - ou termes - ne contiennent pas plus de 2 littéraux, le nombre de clauses résolvantes devient alors polynomial (2-Sat problème). Il en est de même pour le nombre de règles de la base canonique directe qui est exponentiel en fonction du nombre d'éléments, sauf lorsque les règles ne contiennent pas plus de deux éléments. Il s'agit de règles avec un singleton en prémisse et en conclusion (i.e. définies sur $S \times S$).

Par conséquent, lorsque la base canonique est composée de règles ainsi définies sur $S \times S$, le calcul de la base canonique devient polynomial. En effet, on montre facilement que le treillis de fermés d'une telle base est distributif, et que le graphe de dépendance correspond au sous-ordre induit par les sup-irréductibles. La base canonique directe correspond alors à la fermeture transitive du sous-ordre induit par les sup-irréductibles.

Les propriétés structurelles de la base canonique directe permettent de proposer des algorithmes de manipulation adaptés. Citons un algorithme récent de fusion de deux bases canoniques directes [111].

Nous avons proposé une génération incrémentale de la base canonique [26] à partir d'un système implicatif Σ (algorithme 15). Son principe est simple : l'algorithme de transformation d'un système implicatif en sa base canonique directe est exécuté après l'ajout de chaque règle $B \rightarrow x$ de Σ. Ainsi, $B \rightarrow x$ est ajouté dans une base canonique directe, ce qui nous a permis de caractériser de façon précise les règles concernées par l'application récursive du traitement principal de transformation directe, en distinguant les règles $A \rightarrow d$ concernées par une application "droite" - i.e. ajout de $B+A\backslash x \rightarrow d$ - des règles $C \rightarrow e$ concernées par une application "gauche" - i.e. ajout de $C + B \setminus e \rightarrow x$. Le nombre d'implications intermédiaires à générer pour obtenir une base directe se trouve ainsi limité. Bien que cet algorithme ait toujours une complexité exponentielle dans le pire des cas, les expérimentations qui en ont été faites indiquent une amélioration très significative du temps de calcul.

Dans [9] nous proposons une utilisation conjointe des deux bases canoniques. Alors que la base canonique est une représentation minimale et condensée des données pour une meilleure lisibilité, la base canonique directe permet des traitements algorithmiques plus efficaces, que ce soit pour générer un fermé, ou encore pour des ajouts incrémentaux.

4.7 Bibliothèque lattice

Des outils de génération d'un treillis ont tout particulièrement été développés dans le domaine de l'analyse formelle des concepts. Citons en particulier `Toscana` [114], un

Nom : `AjoutDirect`
Data : Une base canonique directe Σ et une implication propre et unaire $B \to x$
Result : La base canonique directe de $\Sigma \cup \{B \to x\}$
début
  //transformation directe;
  Left= $\{C \to e \in \Sigma : e \in B \text{ and } x \notin C\}$
  Right= $\{A \to d \in \Sigma : x \in A \text{ and } d \notin B\}$
  newImpl= $\emptyset$; LeftSet= $\{(\emptyset, \emptyset)\}$
  repeat
    foreach $(X, Y) \in$ *LeftSet* **do**
      delete (X, Y) from LeftSet
      add $X \cup B \backslash Y \to x$ to *newImpl*
      foreach $A \to d \in$ *Right* **do** add $X \cup B \backslash Y \cup A \backslash (Y + x) \to d$ to *newImpl*
      foreach $C \to e \in$ *Left* **do**
        if $C \not\subset X$ *and* $e \notin S$ *and* $e \notin P$ **then** add $(X \cup C, Y + e)$ in LeftSet
  until *LeftSet*$\neq \emptyset$;
  //transformation minimale à gauche;
  foreach $A \to b \in$ *newImpl* **do**
    foreach $C \to b \in \Sigma$ **do**
      if $A \subset C$ **then** delete $C \to b$ from Σ
      if $C \subset A$ **then** delete $A \to b$ from newImpl
  return $\Sigma \cup$ *newImpl*
fin

Algorithme 15: Ajout incrémental d'une nouvelle implication dans une base canonique directe

des premiers outils de génération et de visualisation du treillis des concepts ; `ConExp` [117] développé en java en 2000 qui génère le treillis des concepts d'un contexte et sa base canonique ; `Colibri` [101] qui s'intéresse à une implémentation rapide du treillis des concepts, disponible en Java, C et Caml ; `Galicia` [113] orienté analyse de données, propose plusieurs algorithmes de génération du treillis des concepts. Une étude comparative récente dans [110] fait apparaître que ces outils proposent avant tout des fonctionnalités d'édition d'un contexte, d'édition de graphe, et de visualisation du treillis des concepts. Certains de ces logiciels proposent des algorithmes de visualisation dédiés aux treillis. Cependant, la visualisation de graphes proposée par le logiciel `graphviz` [90] semble fournir des résultats tout à fait satisfaisants.

Dans ce mémoire, nous avons choisi de mettre en avant la notion de système de fermeture, que l'on retrouve à la fois dans un contexte ou dans un système implicatif, ce qui permet d'exploiter de façon générique les liens bijectifs que l'on retrouve entre

treillis, contexte et système d'implications, mais aussi d'unifier l'algorithmique sous-jacente.

Nous avons implémenté en Java une bibliothèque nommée `lattice` *dédiée aux systèmes de fermeture et à leurs treillis de fermés, pour une manipulation fine des objets issus de la théorie des treillis et de leurs propriétés - comme par exemple la manipulation du sous-ordre induit par ses irréductibles, ou la réduction d'un système implicatif - dans un but applicatif expérimental, mais aussi dans un contexte pédagogique d'enseignement par exemple.*

Les figures 4.1 et 4.2 en présentent le diagramme des classes. On y trouve une classe principale et abstraite `ClosureSystem` (figure 4.1) qui contient à la fois les méthodes abstraites spécifiques à un système de fermeture - opération de fermeture et de récupération de l'ensemble des éléments - mais aussi des méthodes génériques de génération - réduction aux sup-irréductibles, génération du treillis de fermés avec l'algorithme nextClosure ou l'algorithme de Bordat.

La classe `Context` permet d'instancier et de manipuler un contexte, alors que la classe `IS` permet d'instancier et de manipuler un système implicatif. Contextes et systèmes d'implications sont tous les deux associés à des systèmes de fermeture, c'est pourquoi ces deux classes `Context` et `IS` héritent toutes les deux de la classe `ClosureSystem`. On trouvera également dans la classe `IS` des méthodes de transformations en un SI équivalent pur, unaire, compact, maximal à droite, minimal à gauche, et en particulier sa base canonique, ou encore sa base canonique directe. Il est possible d'instancier et de sauvegarder un contexte et un système implicatif via un fichier texte de description selon les formats suivants :

Exemple de contexte :

```
Observations: 1 2 3
Attributes: a b c d e
1 : a c
2 : a b
3 : b d e
4 : c e
```

Exemple de système implicatif :

```
a b c d e
a b -> c d
c d -> e
```

La classe `Conceptlattice` (cf figure 4.2), classe support pour un treillis des concepts ou un treillis de fermés, hérite de la classe `Lattice` qui fournit des méthodes spécifiques à un treillis - calcul des irréductibles, d'une borne, de la table, du graphe de dépendance.

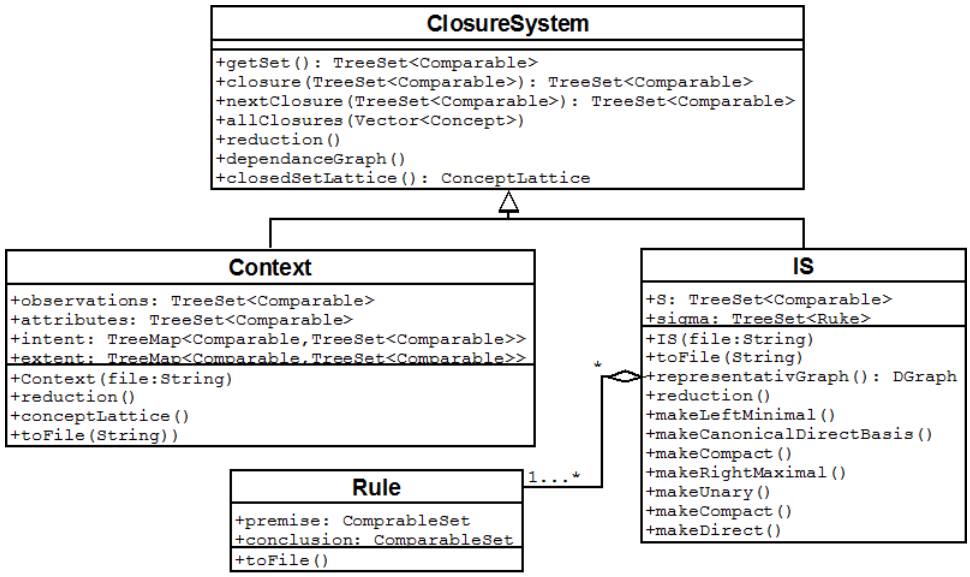


FIGURE 4.1 – Diagramme de classe pour un système de fermeture

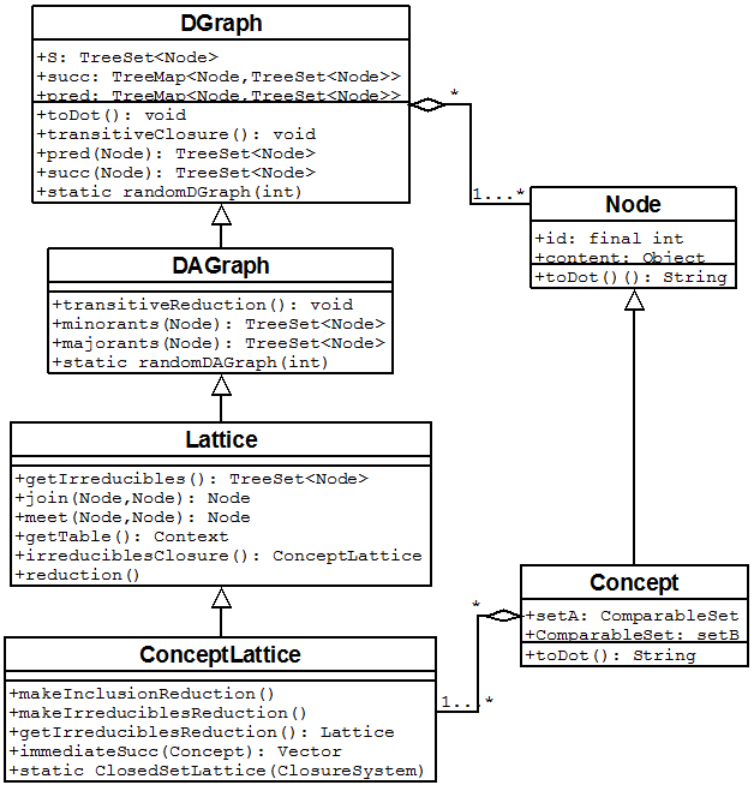


FIGURE 4.2 – Diagramme de classe pour un treillis

La classe `Lattice` hérite quant à elle de la classe `DGraph` qui fournit des méthodes spécifiques à un graphe sans cycles - diagramme de Hasse, tri topologique, minorants, majorants, réduction transitive - et de la classe `DGraph` pour un graphe orienté qui propose notament la fermeture transitive et réflexive, le calcul des composantes fortement connexes, ou encore le calcul d'un sous-graphe. Cette classe implémente un graphe sous forme d'ensembles de successeurs qui encapsule les classes `Node` et `Edge`. Les noeuds d'un treillis des concepts sont des instances de la classe `Concept` qui hérite de la classe `Node`. La classe `Concept` est composée de deux ensembles, avec la possibilité de n'en utiliser qu'un seul pour manipuler un fermé. Tout graphe peut être sauvegardé dans un fichier au format dot, et visualisé à l'aide des outils proposés par le logiciel graphviz [90].

Une étude comparative [97] sur l'impact de la structure de données utilisée pour coder un treillis - tableau de bits, liste chaînée triée, arbre binaire de recherche, table de hachage - montre qu'elles sont similaires. Nous avons choisi de stocker tout ensemble - noeuds, arcs, règles, ou encore concepts - dans un arbre binaire de recherche (collection `TreeSet`) pour ranger et retrouver chaque élément de l'ensemble. Cette classe fournit des méthodes pour les opérations ensemblistes classiques - inclusion, union, intersection, appartenance - et nécessite que les éléments de l'ensemble soit comparables entre eux. La bibliothèque propose également la classe `ComparableSet` pour manipuler des ensembles comparables entre eux par leur contenu (ordre lectique), et ainsi les stocker dans un `TreeSet`, formant ainsi une famille manipulable grâce à des opérations ensemblistes standards.

Une dernière classe `BijectiveComponents` est proposée pour générer automatiquement plusieurs structures à partir d'un système de fermeture quelconque `initialClosureSystem` qui hérite de la classe abstraite `ClosureSystem`. Elle s'instancie de la façon suivante :

```
BijectiveComponents BC = new BijectiveComponents (initialClosureSystem);
BC.initialize();
BC.save(dirString,nameString);
```

L'ensemble des composants suivants sont générés - methode `initialize` - et sauvegardés - méthode `save` - dans des fichiers :

- le treillis de fermés du système de fermeture ;
- la table du treillis ;
- le graphe de dépendance du treillis ;
- la base canonique directe du treillis ;
- les générateurs minimaux du treillis ;
- la base canonique du treillis.

Chacun de ces objets peut ensuite se manipuler de façon plus fine grâce aux méthodes proposées dans les classes associées.

Deuxième partie

Quelques usages pour des données images

Chapitre 5

Introduction

Nos travaux portent sur des *données images*. Les images que nous considérons sont des images graphiques issues de documents. Il s'agit d'images détériorées, et les origines de la détérioration des images sont variées : détérioration du papier (taches, parties effacées), des objets ou des distorsions de numérisation vectorielle dans le cadre des symboles écrits à la main. La problématique de la reconnaissance de symboles issus de documents est une problématique qui a fait l'objet de plusieurs états de l'art dans la littérature [135, 211, 158, 157, 185, 189]. Dans le cadre de projets du laboratoire, nous nous sommes plus particulièrement intéressés à des images de symboles électriques et de lettrines.

La *signature* d'une image ou d'un objet graphique porte une information condensée de l'objet qui permet de le décrire mais aussi surtout de le caractériser dans un objectif de reconnaissance d'images. Une signature se doit, selon le contexte, de respecter certaines contraintes, telles que par exemple l'invariance à la translation, au changement d'échelle ou à l'orientation. De plus, les signatures doivent être le plus possible robustes au bruit. Résultat d'un processus de traitement d'images, la signature d'une image en porte une description qui peut être soit une information globale sur l'organisation des pixels dans l'image (vecteur numérique), soit une information locale sur des primitives issues de l'image (traits, arcs de cercles, ou plus généralement régions de l'image) et leur organisation topologique. La principale difficulté de l'extraction d'une signature réside dans le choix ou la mise en place d'un traitement d'analyse d'images adapté au type des images du domaine.

Signature globale. Une *signature globale* ou statistique porte une information globale sur l'organisation des pixels dans l'image, et peut également caractériser des propriétés fréquentielles. Elle s'organise sous forme d'un vecteur de valeurs numériques nommées *attributs*, *primitives*, *descripteurs* ou encore *caractéristiques*. De bons états de l'art font référence [124, 187, 177, 196]. Dans [187], différentes méthodes d'extraction de primitives y sont décrites, et leurs propriétés d'invariance y sont discutées. Le livre [177] regroupe des articles sur de nombreuses signatures statistiques. Parmi les signatures statistiques classiquement utilisées dans le do-

maine de la reconnaissance de symboles, nous utilisont plus particulièrement la *signature de Radon* [172, 182], les *invariants de Fourier-Melun* [137], la *signature de Zernique* [184], ou encore les *moments de Hue* [145].

Signature locale. Une signature *locale* ou structurelle vise à représenter la structure de la forme, et repose sur une décomposition de la forme en primitives élémentaires ou graphiques et sur leur organisation spatiale ou topologique. Les primitives élémentaires peuvent être des polygones, des segments, des arcs de cercle, et plus généralement des régions de l'image. Leur organisation topologique et spatiale rend la représentation d'un symbole possible : connexion, points de jonctions, parallèlisme, distance, orientation relative, ... Un grand nombre de signatures locales ou structurelles d'objets graphiques a été proposé dans la littérature, on en trouve un bon état de l'art dans [208].

Dans le domaine du document, les problématiques classiquement abordées sont des problématiques de reconnaissance ou de classification d'images de symboles à partir d'une signature, problématiques classiques de la *fouille de données*. Les méthodes proposées nécessitent souvent d'être appliquées sur des images de symboles segmentés, où il s'agit de reconnaître un symbole requête dans une base d'images de symboles pré-segmentés. Cette problématique se distingue de la *recherche par le contenu* (Content Based Information Retrieval), où le symbole requête est à retrouver dans le document lui-même. Par ailleurs, les images de document sont des images issues d'un domaine souvent porteur d'une connaissance métier sémantiquement forte (lettrines, symboles, plan cadastral) qu'il semble naturel d'intégrer à la description bas-niveau de l'image au sein d'un même système. Il s'agit là de problématiques de *représentations des données complexes* décrivant l'image, qui se distinguent de la classification et la reconnaissance d'images.

Les *méthodes de l'AFC* se positionnent à la fois dans des problématiques de fouille de données et de représentation des connaissances, en particulier par des ontologies, problématiques que j'ai abordées dans le cas particulier d'images de documents :

1. Dans un premier temps, mes travaux se sont positionnés dans le domaine de la fouille de données, et plus particulièrement de la *reconnaissance d'images de symboles pré-segmentés*. Ces travaux sont décrit dans le chapitre 6.
2. Dans un second temps, je me suis intéressée aux *relations spatiales* entre primitives graphiques d'une image de document dans un objectif de reconnaissance mais aussi de *représentation de données complexes* issues d'images. Ces travaux sont décrit dans le chapitre 7.

Chapitre 6 : Fouille de données et navigation dans un treillis. La *fouille de données* dans une base d'images d'un domaine particulier s'articule généralement autour de deux étapes que sont l'extraction d'une signature suivie d'une méthode de fouille de données. Une fois les signatures extraites, les principales techniques de fouille de

données, que l'on retrouve dans différents domaines et avec des formulations différentes, que ce soit en statistiques, reconnaissance des formes, apprentissage automatique ou encore analyse de données, ont pour caractéristiques communes à la fois d'analyser des données qui s'organisent classiquement sous forme tabulaire (table objets x attributs) ; mais aussi de répondre à un objectif de description des données par leurs attributs, ou encore de prédiction de nouveaux attributs.

De nombreuses méthodes ont été proposées dans la littérature pour mener à bien ces deux étapes, et le choix de l'approche la plus adaptée au domaine, ainsi que son paramétrage, reste une problématique difficile souvent guidée par les performances expérimentales (taux de reconnaissance, matrice de confusion), la plupart des approches proposées étant des approches de type « boîte noire ». Des approches plus intuitives, parfois qualifiées de symboliques, permettent une meilleure compréhension des données et offrent ainsi l'avantage de pouvoir choisir et adapter les paramètres, mais également les méthodes d'extraction de caractéristiques afin d'obtenir de meilleurs résultats. Ces *méthodes symboliques* se distinguent par des données décrites avec des attributs symboliques ou binaires. Elles nécessitent souvent une phase préalable de discrétisation des données numériques, et permettent ainsi l'intégration de données à la fois numériques et symboliques. Cependant, certaines méthodes permettent de travailler directement sur des données numériques.

Les *règles d'association* et le *treillis des concepts* sont deux approches symboliques, très proches structurellement. La description de données par des règles d'association entre attributs a été particulièrement étudiée dans le domaine des bases de données. Alors que, dans les premières méthodes proposées, les règles extraites sont exhaustives et souvent redondantes, ce qui nuit à leur lisibilité, des stratégies visant à réduire le nombre trop important de règles générées sans perte d'information a mis en avant la propriété de fermeture et le treillis des fermés. Les résultats structurels issus de la théorie des treillis, et en particulier le lien entre les fermés, les générateurs minimaux, et les règles d'implication, sont ainsi exploités. Une restriction des règles générées aux seules règles possédant un attribut de classe en conclusion permet d'envisager leur utilisation en classification supervisée.

Par ailleurs, la problématique de classification est naturellement liée à la *notion de concept* du treillis des concepts, où un concept est un regroupement maximal d'objets possédant des attributs en commun. La classification non supervisée cherche à regrouper des objets ayant des attributs proches, tout en séparant ceux ayant des attributs éloignés, alors qu'en classification supervisée, il s'agit de regrouper des objets ayant une même étiquette (appelée "classe") tout en séparant ceux d'étiquettes différentes. La notion de concept est par conséquent très proche, et le treillis des concepts est naturellement utilisé dans des applications de classification de données supervisées ou non. La plupart des approches de classification basées sur un treillis des concepts sélectionnent des concepts selon un critère de pertinence, avant de les utiliser dans un traitement de

classification de type vote majoritaire.

Contributions et perspectives. *Nous nous sommes intéressés à une utilisation du treillis selon un principe de navigation similaire à celui de l'arbre de décision, et proposons une méthode de classification par navigation dans un treillis. Nommée* NAVIGALA *(NAVIgation into GAlois LAttice), cette méthode est dédiée à la reconnaissance d'objets graphiques détériorés, et plus particulièrement d'images de symboles : à chaque concept correspond un test sur plusieurs variables, la navigation est réalisée à partir du concept minimal, jusqu'à atteindre un concept final représentatif d'une classe. Le verrou principal que nous avons levé ici est un* mécanisme de navigation dans un treillis*, qui généralise celui utilisé dans un arbre de décision. En effet, à chaque étape, les propriétés des treillis induisent une navigation à partir d'une famille de parties d'attributs, alors qu'il s'agit d'un ensemble d'attributs dans l'arbre de décision.*

La stratégie mise en oeuvre induit une différence de comportement vis-à-vis des données par rapport à la stratégie par sélection commune aux méthodes issues de l'AFC qui cherchent à regrouper des attributs fortement dépendant pour une même classe d'objets. La navigation permet d'éviter, dans une certaine mesure, l'influence d'un bruit porté sur plusieurs attributs. En effet, les attributs sont validés les uns à la suite des autres et non par une moyenne, comme dans les méthodes orientées sélection. De plus, l'ensemble des objets est conservé, sans favoriser les plus représentés, à l'inverse des méthodes orientées sélection, ce qui permet d'être plus exhaustif. Enfin, l'ordre de validation des attributs est modifiable en fonction de leur robustesse au bruit, et plusieurs chemins de reconnaissance sont proposés pour une même classe, à la différence de l'arbre de décision qui propose un seul chemin pour atteindre une classe.

Les premièrs résultats obtenus ayant été encourageants [1, 17], c'est dans le cadre des travaux de thèse de Stéphanie Guillas, en collaboration avec Jean-Marc Ogier, que le développement ainsi qu'une étude expérimentale poussée de la méthode NAVIGALA *ont été menés [24, 27]. Nous avons développé le logiciel* NAVIGALA *qui intègre cette méthode au sein d'une chaîne de reconnaissance d'images après une première phase d'extraction de signatures [25]. Nous avons par la suite proposé diverses améliorations. La génération à la demande du treillis permet d'améliorer le coût de génération du treillis et les temps de calcul, tout en garantissant les mêmes temps de reconnaissance [8]. Un système itératif de reconnaissance permettant d'extraire une nouvelle signature lorsque celle utilisée s'avère trop détériorée pour une navigation efficace dans le treillis a également été envisagée [26, 29]. Finalement, une synthèse de ces travaux a été publiée [34], en collaboration avec Muriel Visani.*

Au cours de ces travaux, j'ai pu observer que les treillis générés par la méthode NAVIGALA*, et plus généralement à partir d'une signature numérique discrétisée, forment une classe de treillis particulière que nous avons nommés les* treillis dichotomiques*. Il s'agit d'une classe de treillis intermédiaire entre les modèles hiérarchiques d'arbre et les treillis de manière générale qui possédent des propriétés structurelles fortes, permettant*

ainsi de renforcer le lien entre les treillis et les arbres. En effet, en plus de la stratégie de navigation commune à ces deux structures, tout arbre hiérarchique est inclus dans le treillis [26] pour un même jeu de données discrétisées, à l'inverse, nous avons montré que le treillis est défini comme la fusion de tous les différents arbres hiérarchiques possibles [31, 18]. L'étude des treillis dichotomiques fait l'objet de la thèse en cours de Nathalie Girard, en collaboration avec Muriel Visani. Nous travaillons sur la mise en place d'une structure hybride de classification, intermédiaire entre arbre et treillis, dans le but à la fois d'améliorer les performances de l'arbre et de limiter la trop grande taille du treillis, tout en garantissant la lisibilité de ces structures. Un mécanisme de discrétisation locale a été proposé dans le cadre de ces travaux [23], qui, conjointement avec le mécanisme de génération à la demande, permet d'intégrer dynamiquement de nouveaux attributs au processus de classification, et ainsi d'envisager la mise en place de mécanismes de bouclage de pertinence, et d'adaptativité de la méthode.

Chapitre7 : Représentations spatiales et ontologiques. La difficulté d'extension à des données plus complexes (taxonomie, graphe, ...), mais aussi à une recherche d'information dans les données de plus haut niveau, sémantiquement plus riches, sont les deux principales limites des méthodes de fouille de données. Le domaine du document est principalement concerné. En effet, les images de documents sont des images fortement structurées à partir desquelles il semble naturel d'extraire des primitives (traits, arcs de cercles, régions), et qui sont issues d'un domaine souvent porteur d'une connaissance métier sémantiquement forte.

Dans un objectif de reconnaissance, la comparaison d'images par *appariement de graphes* de primitives reste limitée à des données de faible dimension car il n'existe pas d'algorithme efficace garantissant une solution optimale (problème NP-complet). C'est pourquoi une grande variété d'heuristiques au problème d'isomorphisme a été proposée : il s'agit de techniques dites de "graph probing" à partir desquelles des mesures de similarité entre graphes sont envisageables (distance d'édition, ...) pour mieux comparer les graphes entre eux. Les approches dites de "graph embedings" utilisent, quant à elles, un plongement du graphe dans un vecteur numérique qui constituera la signature structurelle à partir de laquelle des méthodes de reconnaissance plus classiques peuvent s'appliquer. On obtient ainsi une description du graphe qui peut être statistique (voisinage moyen, moyenne des valuations des arcs, ...) mais aussi structurelle basée sur la notion de chemins du graphe représentatifs d'une forme (sacs de chemins, histogramme de chemins).

Alors que l'analyse d'image s'intéresse en particulier à l'extraction de primitives dans un objectif de reconnaissance, leur *représentation spatiale* est une problématique complémentaire abordée en particulier dans les systèmes d'information géographique. En effet, la représentation de l'espace est un sujet de recherche qui suscite l'intérêt à la fois de différentes communautés scientifiques (géographie, linguistique, ...), mais aussi

dans différents domaine en informatique (bases de données spatiales, systèmes d'information géographiques (SIG), raisonnement spatial, bases de données d'images...). Une large gamme de modèles et outils adaptés à la *représentation spatiale* a été développée dans le domaine des systèmes d'informations géographiques (SIG) où les problématiques abordées concernent avant tout une *représentation de données complexes* décrivant des images géographiques, et où l'interopérabilité entre représentations joue un rôle majeur.

Etablir un lien entre une information bas-niveau et une information haut-niveau décrivant des images est une problématique importante dans le domaine l'analyse d'images, identifiée sous le terme de *fossé sémantique*. Les technologies de types *ontologies*, récemment introduites dans le domaine du web sémantique pour permettre d'organiser les données et connaissances d'un domaine dans le but de les partager, diffuser et actualiser, fournissent un cadre méthodologique à la fois pertinent et novateur pour traiter des images de document. Les fondements des ontologies reposent sur les *logiques de description*, famille de langages de représentation des connaissances utilisés pour décrire un domaine applicatif de façon structurée et lisible. Les logiques de description ne sont pas seulement utilisées pour représenter de la connaissance. Elles proposent également des *mécanismes de raisonnement* permettant de déduire de la connaissance implicite à partir de la connaissance explicitement formalisée.

Plusieurs méthodologies de construction d'une ontologie sont proposées dans la littérature, et nécessitent avant tout une bonne compréhension du domaine. Alors que les premières ontologies sont construites à la main, de récents travaux s'intéressent à des méthodes semi-automatiques de construction d'une ontologie directement à partir des données hétérogènes. L'analyse formelle des concepts est une méthodologie d'analyse et de fouille de données particulièrement bien adaptée.

Contributions et perspectives. *Nous avons développé une* signature structurelle robuste *dédiée aux symboles et calculée à partir d'un graphe topologique des segments extraits d'une image de symbole par une adaptation de la transformée de Hough, ce qui lui confère une certaine robustesse vis-à-vis de certaines transformations. Plus précisement, cette signature correspond à un histogramme de chemins du graphe, où un chemin peut s'interpréter comme une forme constitutive du symbole décrivant des agencements spaciaux entre les segments. Il s'agit d'une signature générique car les chemins sont calculés automatiquement à partir de la représentation matricielle du graphe, paramétrables par leur taille. Une classification par navigation à partir d'une description d'un symbole par un ensemble de régions a également été proposée [32, 33] en collaboration avec en collaboration avec Marçal Rusiñol.*

Ces travaux [21, 22] ont été réalisés en collaboration avec Stéphanie Guillas, Simon Bernard, Mickaël Coustaty, Jean-Marc Ogier et Muriel Visani. Une étude expérimentale en a été menée an particulier à l'aide du logiciel Navigala *[30][20] qui s'avère bien adapté. Finalement, une synthèse de ces travaux a été publiée dans un article de revue [19].*

Nous avons par la suite choisi d'explorer la représentation spatiale entre des primitives pour des images de lettrines. Nous avons mis en place une ontologie des lettrines *intégrant à la fois une information sémantique haut-niveau décrivant le contenu des lettrines, mais également une information bas-niveau décrivant des régions extraites des images, ainsi que leur agencement spatial. Les images considérées sont des images de lettrines. Nous avons exploité le mécanisme de raisonnement propre aux logiques de description en proposant une annotation automatique de la sémantique de certaines régions. L'approche proposée est ainsi une contribution à une réduction du fossé sémantique pour des images de lettrines.*

Ces travaux autour des lettrines ont été réalisés en collaboration avec Mickaël Coustaty, Thang Le Ngoc, Alain Bouju et Georges Louis. Il s'agit de travaux récents, un article en en cours de soumission. Plusieurs perspectives sont dores et déjà envisagées, voire amorcées. Nous avons identifié des améliorations de l'ontologie qui nous semblent pertinentes - ajout de nouvelles règles d'inférence, de nouvelles régions, d'une vérité terrain pour une vérification automatique, ...

Nous avons également pu identifier plusieurs caractéristiques propres au mécanisme d'inférence proposé. En effet, il s'agit d'un mécanisme qui peut combiner des informations bas-niveau, mais également des informations sémantiques, le tout de façon itérative. Cependant, il s'agit là de caractéristiques propres à l'approche proposée, qu'il me semble pertinent d'étendre à des images graphiques d'un autre domaine. Des travaux autour d'images de bandes dessinées sont envisagés.

Chapitre 6

Fouille de données et navigation dans un treillis

Les principales méthodes de *fouille de données*, que l'on retrouve dans différents domaines et avec des formulations différentes, que ce soit en statistiques, reconnaissance des formes, apprentissage automatique ou encore analyse de données, ont pour caractéristiques communes à la fois d'analyser des données qui s'organisent classiquement sous forme tabulaire (table objets x attributs) ; mais aussi de répondre à un objectif de description des données par leurs attributs, ou encore de prédiction de nouveaux attributs.

La section 6.1 propose une introduction à la fouille de données et à l'extraction des connaissances. La section 6.2 s'intéresse plus particulièrement aux méthodes de fouille issues de l'AFC. Il s'agit de méthodes symboliques qui se distinguent par des données décrites avec des attributs symboliques ou binaires, et nécessitent une phase de préparation adaptée (section 6.2.1). Elles se positionnent dans un objectif descriptif par des règles d'association (Section 6.2.2) ou encore dans un objectif de classification (Section 6.2.3). La section 6.3 est consacrée à la description de la méthode Navigala (Section 6.3.1), et aux spécificités des treillis dichotomiques qu'elle génère (section 6.3.2). Enfin, des expérimentations de la méthode Navigala sont présentées dans la section 6.4.2.

6.1 Chaîne d'extraction des connaissances

L'extraction des connaissances a pour principal objectif d'extraire dans des grands volumes de données *"des éléments de connaissances non triviaux et nouveaux pouvant avoir un sens et un intérêt pour être réutilisés"*. Elle est composée de plusieurs étapes dont la principale est la fouille de données (cf Figure 6.1). Plusieurs méthodes existent pour mettre en oeuvre la fouille de données. Le choix de l'une d'entre elles est une problématique difficile, en effet, aucune méthode n'est meilleure qu'une autre dans l'absolu. Néanmoins, les propriétés des données, les contraintes, les objectifs et bien sûr les

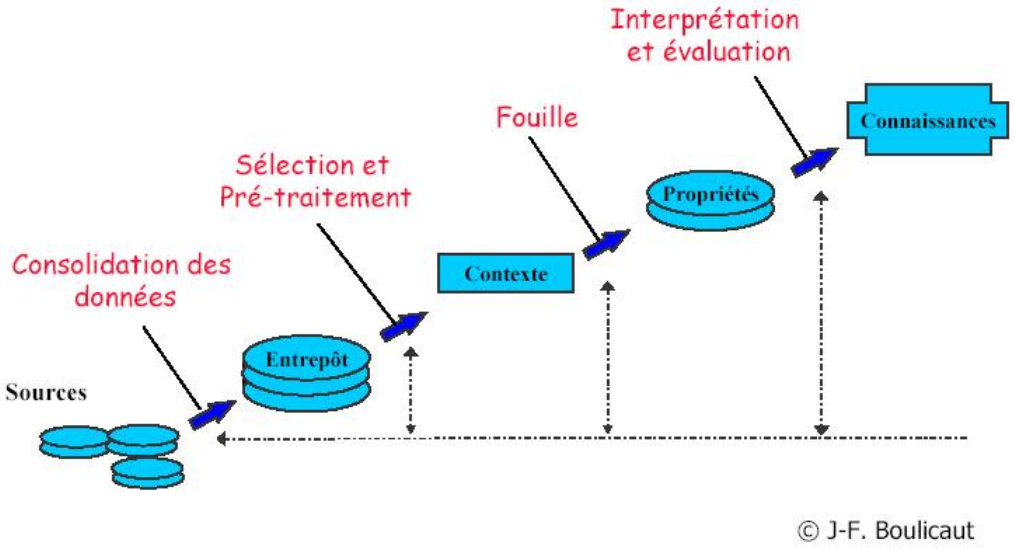


FIGURE 6.1 – Chaîne d'extraction de connaissances

propriétés des méthodes sont des critères de choix importants.

Les méthodes de fouille de données sont implémentées à la fois dans des prototypes de recherche et dans des logiciels commerciaux. Les logiciels du commerce mettent en avant des fonctionnalités d'accès aux données, de coopération de méthodes ou encore d'exploitation des résultats (SAS, Splus, Statistica, SPAD, ..). Les prototypes de recherche s'intéressent davantage à la possibilité d'implémenter de nouvelles méthodes, et de les évaluer dans un contexte expérimental (Tanagra, Weka, R).

En fouille de données, les données sont classiquement décrites par un tableau également appelé table ou matrice. Ce tableau associe à un *objet* (en ligne) un ensemble de valeurs d'*attributs* (en colonnes). On trouve selon les domaines des terminologies différentes : un objet est également appelé enregistrement, individu, observation alors qu'un attribut peut se dénommer caractéristique, variable, champ, ou encore descripteur.

Une distinction est à faire entre des attributs quantitatifs et des attributs qualitatifs. Les *attributs quantitatifs* - ou attributs continus - se définissent sur une échelle continue ou discrète dont la cardinalité du domaine peut être infinie, mais l'écart entre deux valeurs doit être quantifiable, permettant ainsi l'application des opérateurs arithmétiques. Les *attributs qualitatifs* - ou attributs discrets - sont, quant à eux, définis par un nombre de modalités dénombrables, le nombre de valeurs utilisées pouvant être plus ou moins proches du nombre de modalités. On peut faire la distinction entre attributs qualitatifs nominaux pour lesquels les modalités ne peuvent s'ordonner, et les attributs qualitatifs ordinaux pour lesquels il existe une relation d'ordre entre modalités, mais les écarts restent non quantifiables. Les *attributs binaires* sont des attributs discrets à deux modalités.

Le cas supervisé se distingue de par l'existence d'un *attribut de classe* - encore

appellé *étiquette* ou *label* - associé aux données. Il s'agit d'un attribut généralement discret qui représente différentes catégories d'objets, et qu'il s'agit souvent d'inférer pour de nouvelles données.

Une première étape de *préparation des données* est un prétraitement nécessaire pour présenter les données de manière adaptée à la méthode de fouille de données utilisée. La préparation des données regroupe l'intégration des données (regroupement uniformisé de données provenant de plusieurs sources), le nettoyage (gestion des doublons, erreurs de saisie), le traitement des valeurs manquantes, la réduction des données (élimination de redondances, sélection d'attributs pertinents, ..), l'enrichissement des données (ajout de nouveaux attributs par combinaison d'attributs existants), et enfin la transformation des données (normalisation, échelonnage, discrétisation, binarisation) qui dépend avant tout de la méthode de fouille utilisée et peut éventuellement engendrer une perte d'information. En particulier, les méthodes de fouilles issues de l'AFC acceptent généralement des données binaires, et une transformation préalable des attributs numériques (discrétisation, échelonnage) et des attributs discrets (binarisation) s'avère alors souvent nécessaire.

La fouille de données est un traitement consistant à extraire automatiquement des informations depuis les données. Il existe un grand nombre de méthodes de fouilles qui se situent à la frontière entre statistiques et informatique. Certaines méthodes sont issues des statistiques exploratoires pour explorer et décrire les données, ou encore des statistiques décisionnelles. D'autres sont issues de l'informatique : informatique décisionnelle, intelligence artificielle, reconnaissance de forme, ou encore bases de données. Très souvent, ces méthodes reviennent à optimiser les mêmes critères, mais avec des terminologies différentes. On trouve dans la littérature de nombreux ouvrages plus ou moins récents traitant des méthodes de fouilles de données [118, 162, 142, 143, 136]. Les méthodes de fouille diffèrent essentiellement de par l'objectif fixé :

Objectif de description : Il s'agit de décrire au mieux les données dans le but de les réduire ou de les résumer pour une meilleure manipulation (méthodes d'indexation) ; de mettre en valeur des informations présentes mais cachées ; de décrire les associations entre attributs sous forme de règles ; ou encore de regrouper les objets similaires en groupes, classes ou clusters homogènes tout en maintenant les groupes suffisamment larges (on parle de segmentation, de classification non supervisée, ou encore de "clustering").

Objectif de prédiction : Il s'agit de prédire de nouvelles informations à partir des données, ou plus précisément de prédire les valeurs d'un attribut en fonction des autres attributs. On parlera également d'objectif d'explication, l'attribut de classe étant également alors appelé attribut à expliquer, les autres attributs sont alors dits explicatifs. On retrouve les méthodes de classification supervisée ou semi-supervisée, ou encore les méthodes statistiques de régression où l'attribut à expliquer est quantitatif.

La *classification supervisée* est une problématique majeure de la fouille de données qui intègre un traitement préalable d'*apprentissage* pour construire un classifieur décrivant un ensemble de données déjà classées appelé *base d'apprentissage*, suivie d'une phase de *classement* de nouvelles données qui forment la *base de validation*. Eventuellement, une *base de test* peut être utilisée pour adapter le classifieur dans une phase intermédiaire entre apprentissage et classement. En conséquence, le taux de bon ou de mauvais classement permet d'envisager des stratégies simples de validation dans le cadre d'études expérimentales comparatives entre différents classifieurs. La *validation croisée* [149] est la stratégie de validation la plus robuste aux données : l'ensemble des données déjà classées est découpé en n parties, puis la classification est réalisée n fois pour chacune des $n-1$ parties en base d'apprentissage, et la partie restante en classement. La moyenne des n résultats de classification est ainsi indépendante de la répartition des données au sein de chaque base. Une évaluation de méthodes non supervisées de classification selon la même stratégie est envisageable lorsque les données sont déjà classées.

Plusieurs méthodes existent pour mettre en oeuvre la classification, mais aucune n'est meilleure qu'une autre dans l'absolu. On distinguera cependant entre les méthodes numériques qui ne traitent que de données numériques, et les méthodes symboliques qui traitent des données symboliques, mais qui peuvent également intégrer des données numériques via une transformation préalable en prétraitement.

La méthode de classification numérique la plus simple reste la méthode des k plus proches voisins. On peut également citer la méthode des C-moyennes floues, les réseaux de neurones, ou encore les machines à support de vecteurs (SVM) qui semblent être les plus robustes. Parmi les classifieurs symboliques les plus usuels, citons l'*arbre de décision* qui, très étudié dans les années 90 [173, 174], est une référence autant populaire en statistique qu'en apprentissage automatique, et peut intégrer des données numériques par discrétisation. Dans la cadre des travaux de thèse de Stéphanie Guillas [40], nous avons réalisé un comparatif des avantages et inconvénients de méthodes de classification usuelles, dont un récapitulatif est fourni dans les tableaux 6.1 et 6.2. Parmi les méthodes symboliques, celles issues de l'AFC sont en pleine émergence depuis quelques années, et font l'objet de la section suivante.

TABLE 6.1 – Récapitulatif des avantages et inconvénients des différents classifieurs

Classifieurs	Avantages	Inconvénients
Arbre de décision	- technique arrivée à maturité - intuitive et résultats facilement interprétables - pas d'hypothèse a priori sur les distributions des données - gestion des données continues et discrètes - robuste aux données atypiques - intègre un procédé de sélection des attributs	- nombreuses heuristiques impliquent un paramétrage difficile - incapacité à gérer les combinaisons d'attributs - nécessite un ensemble d'apprentissage de taille importante - peut engendrer un fort partitionnement des données - la sélection des attributs peut manquer d'efficacité lorsqu'ils sont en grand nombre et en présence de bruit important - l'ordre de traitement des attributs est figé
C-Moyennes floues	- intuitive et résultats facilement interprétables	- paramétrage : fixer le nombre de classes - dimensionnalité : la distance peut être dominée par des attributs non pertinents
k-PPV	- très bons résultats en général - intuitive et résultats facilement interprétables - robuste aux données bruitées - très simple à mettre en œuvre	- dimensionnalité : la distance peut être dominée par des attributs non pertinents - calcul fastidieux nécessite une méthode d'indexation efficace
Bayésien	- garantit théoriquement l'obtention du taux d'erreur de classification minimal - apprentissage rapide	- suppose que les données suivent une certaine distribution - nécessite un grand ensemble d'apprentissage pour estimer convenablement les probabilités a posteriori

6.2 Méthodes de fouille issues de l'AFC

Plusieurs méthodes issues de l'AFC sont proposées en fouille de données. Il s'agit de méthodes développées dans un objectif de prédiction, ou bien de description, qui exploitent la structure du treillis des concepts, ou bien les règles d'implication qui

TABLE 6.2 – Récapitulatif des avantages et inconvénients des différents classifieurs (suite)

Classifieurs	Avantages	Inconvénients
Réseaux de neurones	- très bons résultats en général - robuste aux données bruitées - prise en compte des combinaisons entre les attributs - aptitude à modéliser des problèmes très variés	- apprentissage long - paramétrage : fixer le nombre de neurones - sensibilité à un trop grand nombre d'attributs non discriminants
SVM	- très bons résultats en général, utilisée comme référence pour situer les autres méthodes - robuste aux données bruitées - pas d'hypothèse a priori sur les distributions des données - intègre un procédé de sélection des attributs - résolution des problèmes non linéairement séparables	- interprétabilité difficile pour les problèmes à grande dimension - grande complexité calculatoire pour traiter de grands ensembles d'apprentissage, mais possibilité de décomposer le problème - paramétrage : choix de la fonction noyau

le décrivent, et qui ont pour caractéristique commune de s'appliquer sur des données binaires. C'est pourquoi une première étape de préparation des données est souvent nécessaire (section 6.2.1). La section 6.2.2 propose une synthèse des méthodes descriptives d'extraction de règles d'association. La section 6.2.3 propose une synthèse de plusieurs stratégies de méthodes prédictives de classification issues de l'AFC.

6.2.1 Préparation des données

Les méthodes issues de l'AFC s'appliquent classiquement sur une table de données binaires, que l'on retrouve également sous la dénomination de *contexte*, où chaque objet est décrit par l'ensemble des attributs binaires qu'il possède. Dans la section 3.1 de la partie I, un contexte est défini par une relation binaire R entre un ensemble d'objets O et un ensemble d'attributs ou d'items I. Les données peuvent également se définir de façon équivalente par une connexion de Galois (α, β) définie entre O et I par :

$$\begin{aligned} \alpha(A) &= \{b \in I \ : \ aRb \text{ pour tout } a \in A\} \\ \beta(B) &= \{a \in O \ : \ aRb \text{ pour tout } b \in B\} \end{aligned}$$

En AFC, la terminologie d'usage pour une table discrète est celle de *contexte multivalué*. Un contexte multivalué y est défini par un quadruplet (O, I, W, R) avec O

l'ensemble des objets, A l'ensemble des attributs, W l'ensemble des modalités des attributs et R une relation ternaire entre O, I et W. $(x, y, w) \in R$, que l'on peut également écrire $y(x) = w$, signifie que l'attribut y a la valeur w pour l'objet x.

La transformation d'attributs discrets en attributs binaires est une opération classique sans perte d'informations que l'on retrouve sous la dénomination de *binarisation* ou encore de *codage disjonctif*. Plus précisément, un attribut discret de n modalités sera transformé en n ou $n-1$ attributs bimodaux ou binaires. Le *codage disjonctif complet* permet de spécifier que $n-1$ données binaires sont créées.

La transformation d'attributs continus en attributs numériques est un traitement plus sophistiqué qui engendre le plus souvent une perte d'information. La *discrétisation* et l'*échelonnage* sont les deux principales techniques de transformation par regroupement des valeurs d'un attribut continu dans des intervalles. La valeur de chaque attribut continu est remplacée par le ou les intervalles la contenant. L'attribut numérique devient ainsi un attribut discret dont les modalités sont des intervalles. Alors que les intervalles issus d'un même attribut seront disjoints après une discrétisation, ils vérifieront une séquence d'inclusion après un échelonnage. Ces transformations induisent le plus souvent une perte d'information, sauf si les intervalles créés reflètent chacun une seule valeur possible d'attribut. Dans ce cas, la non perte d'information est obtenue au détriment du nombre d'intervalles - attributs discrets - créés, et ainsi des performances de la méthode de fouille utilisée.

La *discrétisation* est paramétrable par un *critère de coupe* et un *critère d'arrêt*. Il s'agit de sélectionner, parmi tous les points de coupe possibles entre deux valeurs contiguës d'un même intervalle de valeurs d'une variable, celui qui séparera au mieux les objets. Pour cela, un algorithme standard consiste à tester tous les points de coupe potentiels pour toutes les variables. La variable V et le point de coupe c entre deux valeurs consécutives v_i et v_{i+1} de V maximisant un critère de coupe donné sont retenus. L'intervalle des valeurs de V est alors coupé entre v_i et v_{i+1}, puis remplacé par les deux nouveaux intervalles ainsi créés. Le traitement est réitéré jusqu'à ce qu'un critère d'arrêt soit vérifié, permettant ainsi de stopper la discrétisation. Pour chaque objet, la valeur de chaque attribut est alors remplacée par l'intervalle la contenant.

Il existe de nombreuses méthodes de discrétisation qui sont catégorisées selon trois critères dans [138] :

Mode supervisé ou non supervisé : Dans le mode supervisé, on tient compte du label de classe des objets pour effectuer le découpage, alors que pour le mode non supervisé seules les similarités entre les objets sont considérées. Plus précisément, la discrétisation supervisée cherche à séparer les objets de classes différentes alors que la discrétisation non supervisée vise à regrouper des objets dont les attributs sont similaires.

Mode global ou local : le mode global a lieu en amont de l'application de la méthode de fouille, et tient compte à chaque étape de l'ensemble de tous les objets. Au

contraire, en mode local, le découpage est réalisé au fur et à mesure de l'application de la méthode de fouille qui sélectionne un sous-ensemble d'objets à discrétiser.

Mode statique ou dynamique : le mode statique est la stratégie la plus utilisée. Il s'agit de traiter les attributs indépendamment les uns des autres. A l'inverse, le mode dynamique intègre l'information de tous les attributs simultanément pour construire les intervalles, ce qui permet la prise en compte d'une éventuelle corrélation entre les primitives.

De nombreux critères de coupes sont proposés dans la littérature, qui peuvent être supervisés ou non supervisés selon qu'ils considèrent ou non le label de classe. Parmi les critères de coupe, citons les coupes à intervalles réguliers, les mesures de fréquences entre intervalles, ou encore la distance maximale qui choisit un point de coupe entre les deux valeurs consécutives d'une variable dont la distance est la plus grande. Dans le cas supervisé, les critères les plus usuels sont entre autres les critères de Gini, d'Entropie, du Khi2, de Hoteling, ...

La discrétisation nécessite un second critère, le critère d'arrêt, qui permet de stopper le processus de discrétisation une fois un certain objectif atteint. A nouveau, il existe différents critères d'arrêt. Ainsi la discrétisation peut s'arrêter lorsqu'un nombre préfixé d'intervalles a été construit (mode non supervisé) ou encore lorsque la *séparation des classes* est atteinte (mode supervisé). La séparation des classes sera observée lorsque les objets de classes différentes seront représentés par des ensembles différents d'intervalles. La séparation des classes est un critère classique, qui s'applique naturellement lorsque le critère de coupe est supervisé. En effet, tout critère supervisé s'annule lorsque les classes sont séparées, alors qu'un critère non supervisé peut encore être utilisé. La séparation des classes peut cependant ne pas être atteignable - lorsque deux objets de classes différentes sont décrits pas les mêmes valeurs - où bien peut induire un trop grand nombre d'intervalles. C'est pourquoi le *critère d'homogénéité* ou *critère de pureté* lui est souvent préféré : il s'agit de stopper la discrétisation lorsque les classes sont suffisamment séparées, i.e. lorsque le critère de coupe supervisé est inférieur à un seuil de pureté prédéfini.

Plusieurs comparatifs entre différentes méthodes de discrétisation ont été menés. Citons une étude expérimentale [138] qui montre que les méthodes de discrétisation supervisées permettent d'obtenir de meilleurs résultats. Une comparaison plus théorique [173] établit que discrétisations globale et locale, réalisées par un arbre de décision, possèdent chacune leurs avantages et inconvénients. De façon générale, il apparaît que le choix de la meilleure méthode de discrétisation reste fortement dépendant de l'objectif.

La transformation d'attributs numériques en attributs discrets par *échelonnage* consiste à définir un ou plusieurs *seuils de coupure* pour chaque attribut numérique, seuils qui deviennent les attributs discrets : un objet sera associé à un seuil pour un attribut seulement si sa valeur pour cet attribut est supérieure au seuil. Chaque seuil correspond ainsi à un intervalle de valeurs. Lorsque plusieurs seuils sont définis pour un

même attribut, les intervalles seront alors inclus les uns dans les autres. L'échelonnage est le plus souvent défini dans un mode non-supervisé. Cependant, la catégorisation des différentes méthodes de discrétisation peut également s'appliquer aux différents types d'échelonnage, et le choix du seuil de coupure s'apparente aux critères de coupes.

Divers types d'échelonnage sont proposés. L'échelonnage dit inter-ordinal, introduit dans le cadre de l'AFC dans [58], propose de considérer tous les intervalles de valeurs possibles, ce qui peut induire un très grand nombre d'attributs discrets. Des approches non supervisées d'échelonnage pouvant considèrer des intervalles de plusieurs intervalles ont été proposées dans [75] en utilisant des histogramme, ainsi que dans [64] avec le formalisme des *structures de patrons*.

Le choix d'une technique de transformation d'attributs continus en attributs discrets, ainsi que son paramétrage, est un compromis entre perte d'information et explosion du nombre d'intervalles créés. Cependant, pour la majorité des méthodes issues de l'AFC proposées dans la littérature, il reste difficile d'obtenir des informations précises concernant cette transformation et les critères utilisés - critère de coupe et d'arrêt, seuil de coupure, mode supervisé ou non.

La définition d'un treillis de Galois, plus générale que celle de treillis des concepts, permet d'envisager une description plus sophistiquée des objets par des données non nécessairement binaires (voir section 3.1, partie I). En effet, la connexion de Galois entre les objets et leurs descriptions reste maintenue à condition que soit définie une relation de subsomption sur l'ensemble des descriptions, et la boîte à outils algorithmique de la théorie des treillis est alors utilisable pour ces données. Dans [75], le treillis de Galois est ainsi utilisé pour des objets décrits par des données numériques, ou multivaluées, pour lesquelles une relation de subsomption est définie. Leur objectif est un passage du numérique au symbolique sans passer par une discrétisation des données numériques qui induit une perte d'information. L'*Analyse Logique des Concepts* [55] a été introduite en 2004 comme une généralisation de l'AFC où le treillis de Galois est défini à partir d'un *contexte logique* où chaque objet est décrit par une formule logique. Le formalisme des *structures de patrons*, introduit dans [57] pour des descriptions d'objets par des graphes, également étudiée dans [64] pour des données numériques, est une autre généralisation de ce même principe.

Citons également l'approche par treillis flous [125] qui propose de générer une séquence de treillis $L_1, \ldots, L_n$ pour une suite de valeurs de seuils $s_1, \ldots, s_n$: chaque treillis L_i est construit à partir du contexte C_i obtenu par échelonnage des données de la table binaire selon le seuil s_i. Ainsi, chaque treillis L_i est sous-treillis d'un treillis L_j pour $1 \leq i \leq j \leq n$. Un prétraitement de normalisation des données reste néanmoins nécessaire, et le paramétrage des seuils est une problématique difficile.

6.2.2 Règles d'association et bases

Définitions

Introduite par Agrawal et al. dans [119], l'extraction de règles d'association a été développée pour l'analyse de bases de données. Un exemple souvent cité concerne l'analyse de transactions de vente, afin d'identifier les articles achetés le plus fréquemment ensemble à partir d'une liste d'articles achetés. De façon plus générale, les règles d'association visent à identifier les relations significatives entre des attributs dans un objectif de description, permettant ainsi d'en élargir le champ applicatif à de nombreux domaines. Les mesures d'utilité (le support) et de précision (la confiance) sont utilisées pour ne générer que les associations les plus significatives. Cependant, le nombre d'associations significatives peut potentiellement être élevé.

L'extraction de règles d'association s'applique sur des données qui s'organisent en table binaire ou contexte (O, I, R) porteur d'une connexion de Galois (α, β). C'est pourquoi une première phase de préparation des données est souvent nécessaire. Tout sous-ensemble d'attributs $X \subseteq I$ est également appelé un *itemset*. Le support d'un itemset est défini par :

$$sup(X) = \frac{|\beta(X)|}{|O|} \tag{6.1}$$

Un itemset est dit *fréquent* lorsque son support est supérieur ou égal à un seuil minimal de support *minsupport*.

Une règle d'association est une paire (X, Y), notée $X \rightarrow Y$, composée de deux itemsets $X, Y \subseteq I$, X en est la prémisse, et Y la conclusion, tels que $X \cap Y = \emptyset$. Le support d'une règle d'association correspond au support de l'union des deux itemsets qui la composent :

$$sup(X \rightarrow Y) = sup(X \cup Y) \tag{6.2}$$

La confiance d'une règle $X \rightarrow Y$ mesure la probabilité conditionnelle qu'un objet possède X sachant qu'il possède Y :

$$conf(X \rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)} \tag{6.3}$$

Une règle est dite *valide* lorsque sa confiance est supérieure ou égale à un seuil minimal de confiance $minconfiance$. Une *règle exacte* est une règle dont la confiance est maximale (égale à 1).

Le cas supervisé permet d'introduire un sous-ensemble de règles dites *règles de classification*, ou encore *règles de décision*. Il s'agit de règles dont la conclusion est restreinte à l'attribut de classe, utilisées pour prédire la classe d'un objet en fonction de ses autres attributs.

Algorithmes d'extraction des règles d'association

Le problème de l'extraction des règles d'association d'une table de données binaires consiste à générer l'ensemble des règles dont le support et la confiance sont supérieurs ou égaux à des seuils minimaux de support et de confiance définis par l'utilisateur. L'extraction se réalise classiquement en deux étapes :

1. Extraire la famille $\mathcal{F}$ des itemsets fréquents :

$$\mathcal{F} = \{X \subseteq I \ : \ sup(X) \geq minsupport\} \tag{6.4}$$

2. Générer l'ensemble des règles valides :

$$\{Y \rightarrow X \setminus Y \ : \ X, Y \in \mathcal{F}, \ Y \subset X, \ conf(X \rightarrow Y) \geq minconfiance\} \tag{6.5}$$

   Il s'agit en fait, pour chaque itemset fréquent $X \subseteq I$, de générer l'ensemble des règles valides $Y \rightarrow X \setminus Y$, avec $Y \subseteq X$ itemset fréquent.

La complexité de chacune de ces deux étapes est exponentielle. En effet, le nombre d'itemsets fréquents potentiels est $2^{|I|}$, alors que le nombre de règles valides potentielles pour un itemset fréquent X est $2^{|X|} - 2$. La première étape reste cependant la plus coûteuse en temps. En outre, elle seule nécessite un accès aux données. C'est pourquoi de nombreux travaux se sont intéressés à des méthodes efficaces de génération de l'ensemble des itemsets fréquents.

L'ensemble des parties de I correspond aux $2^{|I|}$ itemsets fréquents potentiels, à l'exception de l'ensemble vide, qui peuvent s'organiser sous forme du treillis booléen des parties de I ordonnées par inclusion. Les deux propriétés suivantes des itemsets fréquents en induisent une caractérisation simple [84] :

1. Tous les sous-ensembles d'un itemset fréquent sont fréquents
2. Tous les sur-ensembles d'un itemset non fréquent sont non fréquents

Les itemsets fréquents correspondent ainsi aux itemsets situés dans le treillis des parties entre l'élément minimal et la *bordure positive*, antichaîne composée des itemsets fréquents maximaux, i.e. des itemsets fréquents tels que tout sur-ensemble n'est pas fréquent. La *bordure négative* correspond quant à elle aux successeurs immédiats des itemsets fréquents maximaux dans le treillis.

Une première stratégie d'extraction des itemsets fréquents consiste à procéder niveau par niveau, à partir de l'élément minimal du treillis, jusqu'à atteindre sa bordure négative. Les premiers algorithmes naïfs d'extraction par niveau ont été proposés en 1993 [119, 144]. L'algorithme fondateur est l'algorithme A Priori [84] qui exploite les propriétés des itemsets fréquents pour déduire les itemsets fréquents d'un niveau $k+1$ par combinaison de ceux calculés au niveau k. On retrouve la même stratégie dans [159]. Par la suite, différentes améliorations ont également été proposées. Dans [84], le temps d'accès aux données est limité en diminuant progressivement la taille de la table binaire.

Dans [179], les données sont partitionnées en n partitions en mémoire, des itemsets fréquents locaux sont calculés pour chaque partition, itemsets locaux qu'il s'agit ensuite de fusionner.

Une seconde stratégie vise à extraire directement les itemsets fréquents formant la bordure positive, puis à déterminer le support de tous les sous-ensembles des itemsets fréquents maximaux. Plusieurs algorithmes ont été proposés. L'algorithme Max-Miner [123], un des plus efficaces expérimentalement, réalise simultanément un parcours du treillis de haut en bas, mais aussi de bas en haut dans le but de limiter les itemsets maximaux candidats.

La plupart des méthodes proposées définissent un ordre total sur les itemsets. Cet ordre total, souvent l'ordre lexicographique, permet à la fois d'éviter des calculs redondants en parcourant chaque itemset au plus une seule fois, mais également de stocker efficacement les itemsets dans des structures de données adaptées.

Fermés et bases de règles d'association

Le nombre d'itemsets fréquents est potentiellement très élevé, d'autant plus quand les données sont denses ou le support minimum faible. La production de données en quantité de plus en plus importante a fait émerger la nécessité de limiter les règles d'association générées. En effet, lorsque le nombre de règles est trop important, les règles sont alors illisibles, inexploitables, et souvent redondantes pour un large nombre d'entre elles.

Une première stratégie de restriction des règles consiste à sélectionner un sous-ensemble des règles générées à l'aide d'indicateurs statistiques de mesures de qualité autres que le support et la confiance. Il existe dans la littérature un grand nombre de mesures de qualité. Ces mesures sont souvent définies en fonction de la probabilité $P(X)$ qu'un objet possède l'itemset $X \subseteq I$. En particulier, la mesure de confiance se redéfinit par :

$$confiance(X \rightarrow Y) = \frac{P(X \cup Y)}{P(X)} \tag{6.6}$$

La *mesure d'intérêt*, étudiée dans [130, 180], intègre la probabilité qu'un objet possède la conclusion de la règle :

$$intert(X \rightarrow Y) = \frac{P(X \cup Y)}{P(X)P(Y)} \tag{6.7}$$

La *mesure de conviction*, définie dans [130], intègre la probabilité qu'un objet possède la prémisse de la règle, mais n'en possède pas la conclusion, mesurant ainsi la dépendance entre présence de la prémisse et absence de la conclusion :

$$conviction(X \rightarrow Y) = \frac{P(X)P(\overline{Y})}{P(X \cup \overline{Y})} \tag{6.8}$$

Une seconde stratégie propose de générer directement un sous-ensemble de règles suffisamment représentatives en exploitant l'opérateur de fermeture $\beta \circ \alpha$ porté par la table binaire des données, et les itemsets fermés qui s'en déduisent (cf section 3 de la partie I). La limitation des itemsets fréquents aux seuls itemsets fréquents et fermés a été proposée dans ce sens en 1999 dans [169]. Elle est rendue possible par le fait que l'ensemble des itemsets fréquents de même fermeture $F \subseteq I$ ont tous le même support $\beta(F)$, et engendrent alors des règles redondantes. On parle de *classe d'équivalence* pour les itemsets fréquents de même fermeture. Le nombre de règles générées à partir des fermés fréquents reste ainsi inférieur au nombre de règles générées à partir de tous les itemset fréquents, sans perte d'information. La restriction du treillis des fermés aux seuls fermés fréquents, appelée *Iceberg de Galois* [181], représente la partie inférieure du treillis qui forme alors un inf-demi-treillis.

Outre les algorithmes classiques de génération du treillis des fermés décrits dans la section 4.4 de la partie I, on trouve dans la littérature des algorithmes dédiés à la génération des seuls fermés fréquents. Citons l'algorithme Close [169, 168], premier algorithme de génération des fermés fréquents, ou encore les algorithmes Pascal [122], LCM [188], Prince [141], ...

Même si les règles générées à partir des seuls fermés fréquents restent limitées en nombre, certaines redondances peuvent subsister. Des travaux se sont alors orientés vers l'extraction d'un sous-ensemble générique de règles d'association, où la généricité se définit plus formellement par les deux propriétés suivantes :

Dérivabilité : possibilité de dériver par un mécanisme d'inférence un ensemble de règles d'association à la fois valide et complet.

Informativité : possibilité de retrouver avec exactitude le support et la confiance des règles dérivées.

Une base générique se définit classiquement en deux parties - la première composée de règles exactes et la seconde de règles approximatives définies à partir des règles exactes - accompagnée d'un mécanisme d'inférence pour déduire l'ensemble des règles d'association, ainsi que leur support et confiance. Le mécanisme d'inférence le plus classique propose d'utiliser l'axiome d'augmentation d'Amstrong [44]. Une étude relativement complète de différentes bases, de leurs propriétés et des différents mécanismes d'inférence a été proposée dans [150, 140]. Les bases construites à partir de la base canonique d'un treillis pour la partie de règles exactes (voir définition 14 section 3.2.2) se révèlent être dérivables mais non informatives. En revanche, deux bases génériques sont introduites dans [80], définies à partir des fermés fréquents et de leurs générateurs minimaux (voir définition 4 Section 2.2.2), sont à la fois dérivables et informatives. Elles sont composées d'une même base de règles exactes GBE (Generic Basis Exact), également appelée base d'implications propres, et d'une base de règles approximatives chacune, GBA (Generic Basis Approximative) et TGBA (Transitive GBA) :

$$GBE = \{X \rightarrow F \setminus X \ : \left(\begin{array}{l} F \text{ fermé fréquent,} \\ X \text{ générateur minimal du treillis} \\ \varphi(X) = F \end{array} \right) \text{et } X \neq F\}$$

$$GBA = \{X \rightarrow F \setminus X \ : \left(\begin{array}{l} F \text{ fermé fréquent,} \\ X \text{ générateur minimal du treillis} \\ \varphi(X) \subset F \\ X \rightarrow F \setminus X \text{ règle valide} \end{array} \right) \text{et } X \neq F\}$$

$$TGBA = \{X \rightarrow F \setminus X \ : \left(\begin{array}{l} F \text{ fermé fréquent,} \\ X \text{ générateur minimal du treillis} \\ \varphi(X) \subset F \\ \varphi(X) \text{ couvert par } F \text{ dans le treillis} \\ X \rightarrow F \setminus X \text{ règle valide} \end{array} \right) \text{et } X \neq F\}$$

La base GBE se compose de règles dont la prémisse est un générateur minimal, et la conclusion est son fermé, règles dont on montre facilement qu'elles sont exactes. L'extension de la base GBE à l'ensemble des itemsets fermés, et non aux seuls fermés fréquents, est équivalente sous sa forme unaire à la base minimale à gauche, ou encore à la base canonique directe du treillis des fermés, (cf section 3.2.2 de la partie I). Les règles approximatives de la base GBA possèdent également un générateur minimal en prémisse, mais la conclusion se forme à partir des fermés successeurs dans le treillis du fermé de la prémisse. Pour limiter le trop grand nombre de règles de la base GBA, la base TGBA se définit par une restriction aux seuls successeurs immédiats du fermé de la prémisse, i.e. en ne considérant que la réduction transitive du treillis. Ainsi, la base TGBA est strictement inclue dans la base GBA. Citons également la base générique informative IGB (Informative Generic Basis) [140] introduite pour limiter l'inconvénient principal de la base générique (GBE,TGBA), à savoir que certaines règles de TGBA\GBA peuvent porter une information "sémantique" nécessaire dans certains contextes d'aide à la décision.

La génération d'une base générique informative s'articule ainsi autour d'une première étape de génération des fermés fréquents - mentionnée ci-dessus - suivie d'une étape d'extraction des générateurs minimaux d'un fermé. L'algorithme d'extraction des générateurs minimaux le plus utilisé [88] est un algorithme incrémental exponentiel - extraction de chaque générateur minimal en temps exponentiel. Il existe également un algorithme incrémental polynomial que nous avons proposé dans la section 4.6 de la partie I, et que l'on retrouve sous une forme différente dans le domaine de la logique [63].

6.2.3 Objectif de classification

Plusieurs méthodes issues de l'AFC sont proposées dans un objectif de classification, où chaque objet est décrit par un ensemble d'attributs ainsi que par une indication de classe introduite sous la forme d'un attribut à prédire. Certaines méthodes proposent une extension des règles d'association à des *règles de classification* possédant l'attribut de classe en conclusion, alors que d'autres exploitent les *concepts* du treillis des concepts. Ces méthodes s'appliquent sur des données binaires, et une phase de transformation de données - discrétisation ou échelonnage, puis binarisation - est alors nécessaire lorsque les données ne sont pas binaires. L'exploitation des bases de règles d'association dans un contexte de classification consiste généralement à intégrer l'attribut de classe aux autres attributs, puis à ne sélectionner que les règles possédant la classe en conclusion. Les méthodes de classification à l'aide du treillis des concepts utilisent l'attribut de classe dans un second temps, après avoir généré le treillis des concepts, pour sélectionner des concepts dits pertinents qui sont ensuite utilisés pour la classification. Une étude récente [161] qui recense quelques méthodes de classification supervisée basées sur un treillis des concepts, apporte des éléments de comparaisons et les résultats y sont semblables voire meilleurs que ceux d'approches plus classiques.

En collaboration avec Stéphanie Guillas dans le cadre de ses travaux de thèse [40], et de Nathalie Girard dans le cadre d'un projet de recherche [39], nous avons réalisé une étude comparative de différentes méthodes de classification issues de l'AFC qui offre un large panorama des différentes stratégies de classification utilisant aussi bien les règles de classification que les concepts du treillis.

Règles de classification : la méthode CBA [156], une des premières méthodes proposées dès 1998, la méthode CMAR [154], les méthodes ARC-AC et ARC-BC [121, 120], la méthode CPAR [191], la méthode Harmony [190], et plus récement la méthode GARCm [128] qui utilise une base de règles informatives.

Classification par sélection de concepts : la méthode GRAND [165], première méthode proposée, la méthode LEGAL [155], son extension [160], la méthode GALOIS [133], la méthode RULEARNER [178], la méthode CIBLe [163], et les méthodes CLNN et CLNB [193].

Dans les sections suivantes, une synthèse de cette étude comparative est proposée, faisant apparaître les différentes stratégies mises en oeuvre dans les méthodes étudiées, méthodes qui s'articulent classiquement autour d'une phase d'apprentissage, suivie d'une phase de classement.

Règles de classification

Apprentissage. L'apprentissage consiste à extraire des règles de classification à partir des données. Pour cela, l'attribut de classe est intégré à la table de données.

L'extraction est ensuite réalisée en deux étapes : une étape de génération des règles d'association, suivie d'une étape d'élagage.

Dans les premières méthodes proposées (méthodes CBA, CMAR, ARC, CPAR), l'extraction des règles d'association est réalisée à l'aide de l'algorithme Apriori ou d'une de ses variantes. La méthode CMAR utilise ainsi la variante FP-growth qui optimise l'espace mémoire en stockant les règles dans une structure de données arborescente, alors que la méthode CPAR génère directement les règles de classification pertinentes par intégration d'une fonction de gain à maximiser au cours de l'extraction. Par la suite, des méthodes exploitant l'Iceberg des fermés fréquents (méthode Harmony), ou encore une base générique de règles (méthode GARCM) ont été proposées. Une fois les règles d'association générées, il s'agit d'en extraire les *règles de classification*. Seules les règles possédant un attribut de classe en conclusion sont conservées.

L'élagage est indispensable lorsque les règles extraites sont abondantes et redondantes, ce qui est le cas des règles extraites par l'algorithme APriori ou une de ses variantes. A l'inverse, la phase d'élagage n'a plus lieu d'être dans les méthodes plus récentes qui extraient une base générique de règles. De façon similaire au cas non supervisé, une première stratégie d'élagage consiste à utiliser des mesures statistiques de la qualité des règles pour sélectionner certaines règles : corrélation minimale entre les règles basée sur une métrique du χ^2 (méthode CMAR) ou encore selon un critère de précision (méthode CPAR).

L'exploitation de l'indicateur de classe dans le cas supervisé permet cependant de proposer des stratégies d'élagage mieux adaptées. Ainsi, les méthodes CBA et ARC réalisent un tri des règles de façon à ce que la première règle classant correctement tout objet de la base d'apprentissage précède la première règle qui le classe mal. Un premier tri par précédence dans la méthode (CBA) - ordre décroissant de la confiance, et en cas d'égalité, du support - ou par précision dans la méthode ARC - prémisse de cardinalité croissante - permet d'initialiser le traitement. La méthode CBA réalise ensuite un second élagage basé sur le taux d'erreur : lorsque l'ajout d'une nouvelle règle fait baisser le taux de reconnaissance, alors cette règle et les suivantes ne seront pas conservées. La méthode Harmony, qui exploite les propriétés des itemsets fermés, propose quant à elle un élagage des attributs des itemsets fermés fréquents avant d'en extraire les règles.

Classification. La classification consiste à déduire la classe d'un objet requête de l'application des règles. Un objet vérifie une règle lorsqu'il en possède les attributs en prémisse. Un mode de classement simple consiste à associer à l'objet requête la classe conclusion de la première règle vérifiée (méthode CBA), ce qui nécessite que les règles aient été préalablement triées en apprentissage - tri par précédence, ou par précision. Cependant, les règles ne sont alors pas toutes exploitées. C'est pourquoi un mode de classement utilisant l'ensemble des règles vérifiées par l'ob-

jet requête, au lieu de la première règle vérifiée seulement, a été préféré dans les autres méthodes. Les règles vérifiées sont regroupées par classe, et le nouvel objet sera affecté à la classe du regroupement de règles qui, pour une mesure de qualité donnée, possède : la règle dont la mesure est la meilleure ; la meilleure mesure moyenne des k meilleures règles du paquet ; ou encore la meilleure mesure moyenne des règles du paquet. Différentes mesures de qualité sont utilisées dans les méthodes étudiées : la confiance (méthodes ARC et Harmony), la précision (CPAR), la mesure pondérée du χ^2 (méthode CMAR), l'intérêt (méthode GARCm), ou encore la surprise (méthode GARCm).

Classification par sélection de concepts

Apprentissage. Dans un premier temps, les concepts du treillis sont générés en mode non supervisé à partir des données binaires non classées à l'aide d'un algorithme classique de génération de treillis (voir Section 4.4), d'un algorithme incrémental (méthodes Galois, Rulerarner), ou encore d'un algorithme de génération de l'Iceberg de Galois [181] pour n'obtenir que les concepts fréquents, i.e. composés d'un nombre minimal d'objets (méthode Grand).

Dans un second temps, des concepts suffisamment représentatifs pour la classification sont sélectionnés. La plupart des méthodes utilise un critère supervisé d'homogénéité pour sélectionner des concepts suffisamment homogènes pour chaque classe (méthodes Grand, Galois). La méthode Rulearner ne retient que les concepts homogènes maximaux pour chaque classe, i.e. composé d'un ensemble minimal d'attributs et maximal d'objets.

Les concepts ainsi sélectionnés sont alors labélisés par leur classe majoritaire, et forment le classifieur dans la plupart des méthodes : représentation par un ensemble de concepts pertinents par classe (méthodes Legal, Galois) ou encore par un ensemble de règles où chaque règle est composée en prémisse des attributs des concepts pertinents, et en conclusion du label de classe du concept (méthodes Grand, Rulearner). Dans la méthode Cible, les attributs des concepts pertinents sont utilisés pour sélectionner des *objets prototypes* - objets possédant les attributs pertinents - qui serviront de classifieurs. Dans les méthodes CLNN et CLNB, les attributs pertinents servent à construire des *règles contextuelles* de classification. Une règle contextuelle permet de sélectionner un sous-ensemble d'attributs sur lesquels appliquer le classifieur du plus proche voisin pour la méthode CLNN (NN pour Nearest neighbor), et le classifieur bayésien naïf pour CLNB (NB pour Naive Bayes).

Classification. Une fois le classifieur construit, il s'utilise pour classer - prédire l'attribut de classe - d'un objet requête sur la base de ses attributs. Lorsque le classifieur se compose d'un ensemble de concepts pertinents labélisés par une classe, ou de façon similaire par des règles avec une classe en conclusion, la classification d'un

objet requête consiste à vérifier si l'ensemble de ses attributs contient ceux de chaque concept pertinent, ou bien ceux de la prémisse de chaque règle, ce qui permet d'identifier un ensemble de concepts ou de règles vérifiés par l'objet requête. Les concepts et règles vérifiés servent ensuite au classement : la classe affectée à l'objet peut être soit la classe majoritaire parmi tous les concepts ou règles vérifiés ; ou encore la classe du concept ou de la règle la plus similaire, la similarité étant donnée par le nombre d'attributs du concept ou de la règle partagée par l'objet. Dans la méthode Cible, le classifieur se compose d'objets prototypes, et la classe affectée à l'objet sera la classe majoritaire parmi celle des k plus proches objets prototypes. Dans les méthodes CLNN et CLNB, la classification est réalisée en sélectionnant les règles contextuelles vérifiées par l'objet requête. La classe affectée à l'objet est la classe majoritairement renvoyée par le classifieur (NN ou NB) appliqué sur les seuls attributs en prémisse des règles vérifiées.

Comparatif des méthodes

Les tableaux 6.3 et 6.4 proposent un récapitulatif des méthodes à base de règles. En conclusion, les premières méthodes à bases de règles (méthodes CBA, CMAR, ARC) sont exhaustives car elles génèrent toutes les règles valides avant de n'en sélectionner que certaines. Les méthodes plus récentes cherchent à limiter l'ensemble des règles générées en utilisant des heuristiques (méthodes CPAR, Harmony) ou encore en exploitant les propriétés des bases de règles génériques (méthode GARCm), permettant ainsi de diminuer de manière conséquente les temps d'exécution.

Le tableau 6.5 propose une synthèse détaillée des méthodes utilisant le treillis, en indiquant le type d'apprentissage proposé (S=supervisé, NS=non supervisé). La figure 6.2 permet de situer ces méthodes. Elles peuvent être regroupées selon qu'elles utilisent seulement les concepts, les concepts et les règles, une double sélection, ou encore des règles contextuelles. Ces méthodes se résument à une sélection de concepts, dans le but d'identifier des groupes d'attributs fortement dépendants pour une même classe d'objets. Une construction incrémentale du treillis est envisageable dans le but de réduire du temps de traitement. Ainsi, il n'est pas nécessaire de reconstruire tout le treillis pour prendre en compte de nouveaux objets.

Ces méthodes de classification issues de l'AFC ont été testées sur des bases différentes, et il reste difficile d'en établir un comparatif expérimental exhaustif. Les résultats expérimentaux mentionnés font cependant apparaître des taux de classification meilleurs que des classifieurs symboliques usuels que sont l'arbre de décision (méthodes C4.5, CBA) ou le classifieur bayésien, et comparables à des classifieurs numériques (méthode SVM). Ces méthodes varient selon plusieurs critères dont l'étape de discrétisation, mais aussi le treillis obtenu (souvent un treillis réduit à partir d'un critère de validité des concepts), ou encore les algorithmes de classification utilisés. Leur principal inconvénient reste le temps de calcul inhérent à la construction du treillis.

TABLE 6.3 – Propriétés des méthodes de classification basées sur les règles d'association

Méthodes	Génération des règles	Elagage des règles
CBA	Algorithme Apriori Support minimal Exhaustivité des règles	Oui, par le taux d'erreur pessimiste
CMAR	Basée sur FP-growth Support et confiance minimaux Exhaustive	Oui, par précédence, par corrélation positive et par couverture de l'ensemble d'apprentissage
ARC	Extension d'Apriori (FP-Growth) Exhaustivité des règles	Oui, par précédence des règles spécifiques et par couverture de l'ensemble d'apprentissage
CPAR	Extension A Priori (heuristique de gain minimal) Non exhaustivité des règles	Non, sélection réalisée en amont sur les motifs
HARMONY	Tri des motifs fréquents, méthode diviser pour régner et recherche en profondeur Support minimal, élagage des motifs de support équivalents, élagage des attributs et des motifs non prometteurs Non exhaustive	Non, sélection réalisée en amont sur les motifs
GARCm	Règles de la base $\mathcal{IGB}$	Oui, par couverture de l'ensemble d'apprentissage

6.3 Méthode NAVIGALA de classification par navigation

La synthèse des méthodes de classification issues de l'AFC fait apparaître qu'elles cherchent avant tout à réaliser des regroupements d'attributs fortement dépendants pour une même classe d'objets, en exploitant les règles d'association ou encore les concepts du treillis des concepts.

Des stratégies reposant sur la structure du treillis elle-même sont également envisageables, où il s'agit de l'utiliser comme un espace de recherche et de navigation de manière similaire à celle de l'arbre de décision. Intuitivement, en partant du concept

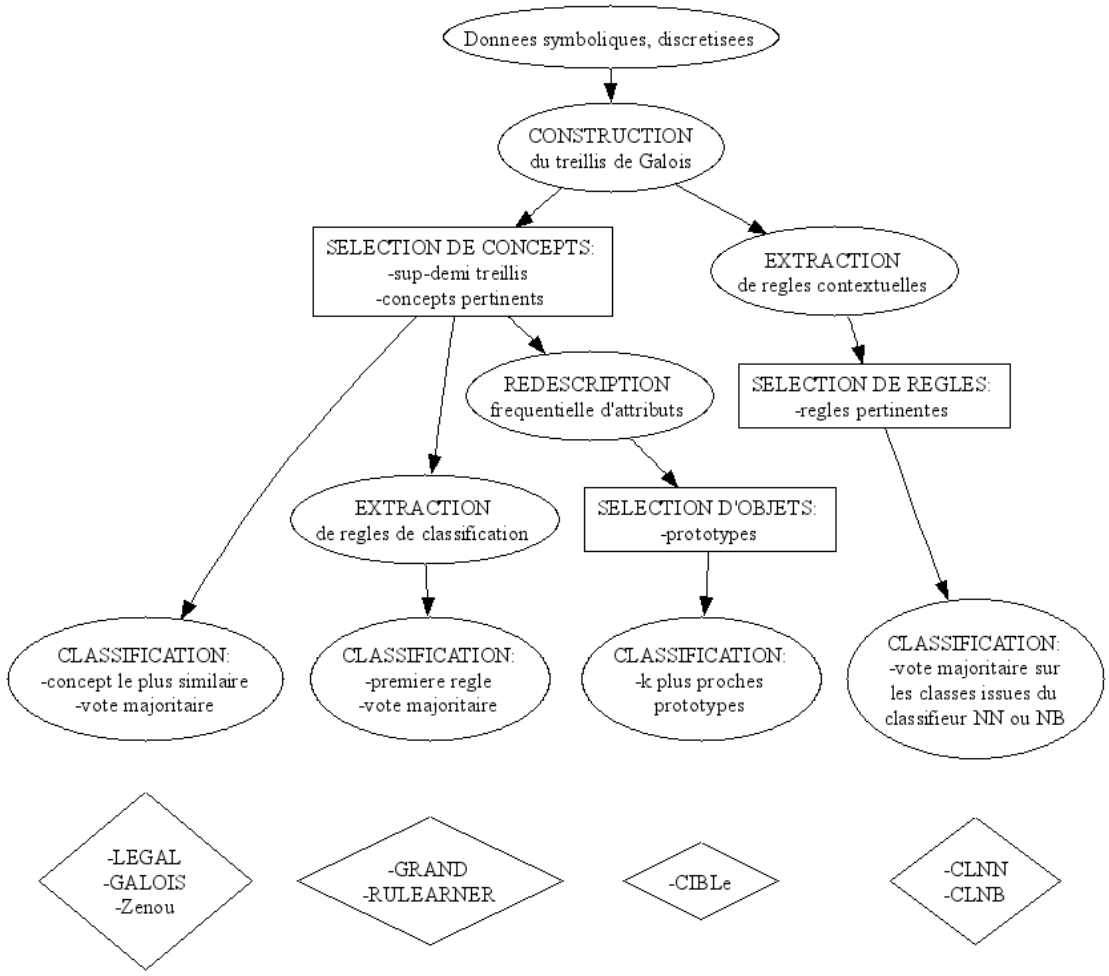
TABLE 6.4 – Propriétés des méthodes de classification basées sur les règles d'association (suite)

Méthodes	Tri des règles	Classification
CBA	Oui, par précédence et par erreur de classification de l'ensemble d'apprentissage	La première règle vérifiée par l'objet
CMAR	Oui, par classe	Ensemble de règles vérifiées par classe obtenant le meilleur χ^2 pondéré
ARC-AC et ARC-BC	Oui, par classe et par précédence	Ensemble de règles vérifiées par classe obtenant la meilleure confiance moyenne
CPAR	Oui, par classe et par estimation de l'erreur de Laplace (pouvoir de prédiction)	Ensemble des k premières règles vérifiées par classe obtenant la meilleure estimation de Laplace
HARMONY	Oui, par classe et par précédence	Ensemble de règles vérifiées par classe obtenant la meilleure confiance moyenne
GARCm	Non	La règle vérifiée qui obtient le meilleur indice de qualité

minimal, et en progressant vers le concept maximal, on observe une spécialisation de l'ensemble des objets et une généralisation de l'ensemble des attributs, ce qui se traduit pas une réduction du nombre d'objets au fur et à mesure que le nombre d'attributs augmente. Plusieurs méthodes ont été proposées en ce sens dans un objectif de recherche d'information [134], souvent en interaction avec l'utilisateur qui peut ainsi observer les dépendances entre attributs.

Nous proposons une nouvelle méthode de classification par navigation dans le treillis des concepts, nommée NAVIGALA (NAVIgation into GAlois LAttice). Ce principe de classification par navigation a été initié dans le cadre de travaux de stage recherche [35, 41], en collaboration avec Jean-Marc Ogier, les résultats encourageants obtenus ont donné lieu à des publications [1, 17]. C'est dans le cadre des travaux de thèse de Stéphanie Guillas [40] que nous avons développé la méthode NAVIGALA, dans un objectif

FIGURE 6.2 – Points communs / différences des méthodes de classification par sélection de concepts issues de l'AFC



de reconnaissance d'objets graphiques détériorés, et plus particulièrement d'images de symboles. Le verrou principal que nous avons levé ici est de proposer un mécanisme de navigation dans un treillis similaire à celui proposé dans un arbre de décision. En effet, le treillis propose à chaque étape de navigation non pas un ensemble d'attributs, mais une famille de parties d'attributs.

Ces travaux ont fait l'objet de plusieurs articles de conférences qui décrivent les premiers résultats [24, 27] ; le mécanisme de génération à la demande permettant de réduire le coût de génération du treillis tout en garantissant les mêmes taux de reconnaissance [8] ; un système itératif de reconnaissance [26, 29]. Une synthèse a été publiée récemment dans un article de revue [34], en collaboration avec Muriel Visani. La méthode NAVIGALA a également été implémentée dans un logiciel du même nom dans le cadre d'un stage recherche [43]. Un mécanisme de discrétisation locale a récemment été

TABLE 6.5 – Propriétés des méthodes de classification basées sur les treillis de Galois

Méthodes	Apprentissage	Construction	Classification
GRAND	S et NS	Treillis Sélection des concepts les plus généraux de chaque classe Extraction des règles de classification	Vote majoritaire parmi les règles les plus spécifiques pour l'objet
LEGAL	S	Sup-demi treillis contenant les concepts pertinents	Vote majoritaire parmi les concepts pertinents
GALOIS	S et NS	Treillis Sélection des concepts pertinents	Concept le plus similaire pour l'objet ou vote majoritaire
RULE-ARNER	S	Treillis Sélection des concepts pertinents Extraction des règles de classification	Application de la première règle vérifiée par l'objet à classer ou vote majoritaire parmi les règles vérifiées
CIBLe	S	Sup-demi treillis Sélection des concepts pertinents Redescription du contexte Sélection de prototypes	Vote majoritaire parmi les plus proches voisins
CLNN et CLNB	S	Treillis Extraction des règles contextuelles Sélection des règles pertinentes	Vote majoritaire parmi les classes retournées par le classifieur NN ou NB des règles contextuelles vérifiées par l'objet

proposé en collaboration avec Nathalie Girard dans le cadre de sa thèse en cours [23].

Les treillis générés par la méthode NAVIGALA à partir d'une signature numérique discrétisée, ou plus généralement à partir d'un vecteur numérique de taille fixée, forment une classe de treillis intermédiaire entre les modèles hiérarchiques d'arbre et les treillis de manière générale, que nous avons baptisés les *treillis dichotomiques* dans [26]. Il s'agit de treillis aux propriétés structurelles fortes qui peuvent s'interpréter comme la fusion de tous les différents arbres de décision possibles [31, 18]. L'étude des treillis dichotomiques fait l'objet de la thèse en cours de Nathalie Girard.

Cette section est consacrée à la description de la méthode NAVIGALA (section 6.3.1) et des treillis dichotomiques (section 6.3.2).

6.3.1 Description de la méthode NAVIGALA

Développée dans un objectif de reconnaissance d'images de symboles, la méthode NAVIGALA traite des signatures d'images, et plus généralement des vecteurs de données numériques. Chaque image - ou objet - est décrit par sa signature - un vecteur de valeurs numériques - ainsi que par une identification de classe - un attribut discret - indiquant la classe de l'image. La reconnaissance s'inscrit dans un processus de classification supervisée qui s'articule classiquement autour de trois phases : préparation des données ; apprentissage ; classement.

Préparation des données

En classification supervisée, l'ensemble des données se décompose en deux bases distinctes : une *base d'apprentissage* de données déjà classées à partir desquelles le classifieur est généré ; une *base de validation* de données à classer. Ainsi, la préparation des données sera différente selon que les objets appartiennent à la base d'apprentissage où à la base de validation. En effet, les données numériques de la base d'apprentissage nécessitent d'être transformées en données binaires pour construire le treillis des concepts. Par ailleurs, l'attribut de classe est utilisé. En revanche, il n'est pas nécessaire de binariser les données numériques de la base de validation, et l'attribut de classe n'est utilisé que pour vérifier si le classement est correct ou non. Cependant, la stratégie de validation par validation croisée nécessite que chaque objet apparaisse successivement à la fois en base d'apprentissage et en base de test. C'est pourquoi le prétraitement des données sera réalisé sur l'ensemble des données.

Pour que les données soient comparables entre elles, il est nécessaire qu'elles soient tout d'abord mises dans un même ordre de grandeur. La normalisation des données est une transformation classique d'uniformisation des échelles et d'harmonisation des données. La simple normalisation linéaire consiste à recaler l'ensemble des valeurs dans un nouvel intervalle de données. La prise en compte de la distribution des valeurs permet d'envisager une normalisation qui offre des propriétés intéressantes telle que la symétrie des données, mais nécessite de faire une hypothèse sur la loi de distribution. La plus classique reste la normalisation selon une loi linéaire, qui intègre la moyenne et l'écart-type des valeurs. NAVIGALA propose une normalisation des vecteurs de caractéristiques suivant un loi linéaire et une loi normale.

Les données sont initialement décrites par une table de données numériques où chaque colonne correspond à une caractéristique $i \in I$, chaque ligne correspond à un objet $x \in O$ décrit par son vecteur de caractéristiques normalisé $(f_1, \ldots f_n)$. Une phase de discrétisation permet de partitionner les valeurs des attributs en intervalles disjoints qui forment ainsi un ensemble d'attributs discrets. A l'issue de cette discrétisation, on obtient une table de données discrètes où chaque objet $x \in O$ est décrit par un vecteur d'intervalles $(v_1, \ldots, v_n)$, chaque intervalle v_i contient la valeur initiale f_i de

la caractéristique i de l'objet. Une binarisation permet alors d'obtenir des attributs binaires qui s'organisent en contexte (O, I, R) où chaque objet $x \in O$ est en relation selon R avec un ensemble d'attributs discrets, chaque attribut discret $v \in I$ est un intervalle issu d'une caractéristique i qui contient la valeur initiale f_i de l'objet. Les caractéristiques non discrétisées au cours du traitement - i.e. dont les valeurs couvrent un seul intervalle - ne sont pas intégrées dans la table binaire, conférant ainsi à la méthode une propriété de sélection des caractéristiques. La connexion de Galois (α, β) portée par tout contexte (O, I, R) est la base de l'existence du treillis des concepts.

Dans NAVIGALA, plusieurs types de discrétisation sont proposés, qui diffèrent selon les critères utilisés :

Critères de coupe : le critère de la *distance maximale* est un critère de coupe non supervisé qui consiste à rechercher l'intervalle qui possède l'écart maximal entre deux valeurs consécutives lorsque ces valeurs sont triées par ordre croissant. Les deux autres critères proposés sont des critères supervisés qui tiennent compte de l'attribut de classe des objets. La *fonction d'entropie* est une mesure caractérisant le degré de mélange des classes. Enfin, le *coefficient de Hotelling* tient compte à la fois des distances entre les valeurs et des labels de classes pour maximiser la distance entre les classes, et également minimiser la dispersion des classes.

Critère d'arrêt : le critère d'arrêt proposé par NAVIGALA est un critère supervisé d'homogénéité défini par un *seuil de pureté*. La discrétisation s'arrête lorsque les classes sont suffisamment pures, selon le seuil fixé. Ce critère correspond à la séparation des classes lorsque le seuil est à 0, mais permet d'envisager une discrétisation de données non séparables - i.e. lorsque des objets de classes différentes ont les mêmes caractéristiques - lorsque le seuil est strictement positif. A nouveau, la fonction d'entropie ou le coefficient d'Hotelling sont utilisés pour mesurer la pureté des classes : plus la valeur est proche de 0, plus les classes sont homogènes.

La discrétisation est proposée en *mode global* ou en *mode local*. En mode global, la discrétisation considère l'ensemble des données pour sélectionner à chaque étape, parmi tous les points de coupe possibles entre deux valeurs contigües d'un même intervalle de valeurs d'une variable, celui qui séparera au mieux les objets. Rappelons que tous les points de coupe potentiels pour toutes les variables - i.e. intervalles de valeurs - sont testés, et la variable V et le point de coupe c entre deux valeurs consécutives de V maximisant un critère de coupe donné sont retenus. Le traitement est réitéré jusqu'à ce qu'un critère d'arrêt soit vérifié, permettant ainsi de stopper la discrétisation. La discrétisation globale est ainsi réalisée indépendamment de la méthode de classification utilisée. A l'inverse, une discrétisation des données en mode local consiste à choisir la segmentation qui permettra de discriminer au mieux les objets d'un sous-ensemble qu'il s'agit d'identifier selon la méthode de fouille utilisée. La plupart des méthodes construisant un arbre de décision proposent ainsi une discrétisation locale en sélectionnant à chaque étape de construction un noeud de l'arbre dont les objets sont à discriminer.

En collaboration avec Nathalie Girard, nous avons récemment proposé un algorithme de *discrétisation locale* pour la méthode NAVIGALA. Chaque concept non homogène (A, B) du treillis des concepts porte une information qui permet à la fois d'identifier le sous-ensemble d'objets A à discriminer, ainsi que le sous-ensemble B d'intervalles potentiels à discrétiser pour les discriminer. Il s'agit alors de déterminer l'intervalle $V((A, B))$ de B dont le meilleur point de coupe c permet de maximiser le critère de coupe $gain(V, c(A, B))$ sur les seuls objets de A.

La génération du treillis étant très coûteuse, il n'est pas envisageable de générer l'ensemble des concepts du treillis à chaque étape de discrétisation pour sélectionner ceux qu'il convient de discriminer. Les propriétés structurelles d'un treillis nous ont permis d'identifier précisément certains concepts non homogènes à discrétiser. En effet, lorsque le critère d'arrêt n'est pas vérifié, alors il existe un ensemble d'objets à discriminer, et décrits par le même ensemble d'intervalles. Ces objets sont donc équivalents dans la table binaire où les attributs sont les intervalles déjà créés, et appartiendront ainsi aux mêmes concepts du treillis. En particulier, ils appartiennent au concept maximal les contenant qui, par définition, est un *concepts inf-irréductibles* du treillis, concepts particuliers définis dans la section 4.3 de la partie I. Par conséquent, lorsque le critère d'arrêt n'est pas vérifié, alors il existe un concept inf-irréductible (A, B) non homogène. Les propriétés structurelles d'un treillis permettent de générer directement ses seuls concepts inf-irréductibles à partir de la table binaire. La section 4.3 de la partie I est consacrée à ce problème algorithmique de génération des inf-irréductibles.

Considérons, pour une étape de discrétisation, l'ensemble M des concepts inf-irréductibles non homogènes. Dans un premier temps, nous utilisons le critère de coupe pour calculer, pour chaque concept $(A, B) \in M$, l'intervalle $V((A, B))$, et son point de coupe c, candidat à la discrétisation. Soit I_M l'ensemble contenant, pour chaque concept $(A, B) \in M$, l'intervalle de B candidat à la discrétisation. L'intervalle V_M^* à discrétiser est ensuite sélectionné parmi I_M, deux modes de sélection sont proposés :

Discrétisation locale simple :

$$V_M^* = argmax_{V(A,B)\in I_M}(gain(V, c, (A, B)) \tag{6.9}$$

Discrétisation locale linéaire :

$$V_M^* = argmax_{V(A,B)\in I_M}(\sum_{(A',B')\in M, V\in B'} gain(V, c, (A', B')) \tag{6.10}$$

Alors que la discrétisation locale simple sélectionne l'intervalle $V(A, B)$ qui permet d'obtenir le meilleur gain sur les objets A, la discrétisation locale linéaire intègre également l'impact du gain sur les objets A' des autres concepts inf-irréductible (A', B') de M contenant l'intervalle. L'intervalle sélectionné V_M^* est alors coupé en deux selon son point de coupe c, puis remplacé par les deux intervalles ainsi créés. L'ensemble des concepts inf-irréductibles du treillis est ensuite recalculé directement à partir de cette

nouvelle table binaire, et seuls les concepts inf-irréductibles non homogènes sont stockés dans M. le processus est réitéré jusqu'à ce que tous les concepts inf-irréductibles vérifient le critère d'arrêt.

Ce processus est initialisé en créant, pour chaque caractéristique de la table initiale de données continues, un intervalle - ou attribut - contenant l'ensemble des valeurs observées. La table discrète ainsi composée d'intervalles est binarisée, et son treillis des concepts contient alors un unique concept avec tous les objets et tous les intervalles. Ce concept est un concept inf-irréductible du treillis qui, clairement, n'est pas homogène. Il est utilisé pour initialiser l'ensemble M.

NAVIGALA propose ensuite d'étendre chaque intervalle issu du traitement de discrétisation en un *nombre flou*, ce qui permet d'affiner l'appartenance d'une valeur dans l'intervalle au cours de la phase de classification, et ainsi d'en améliorer la robustesse. En effet, le taux d'appartenance $\mu(x)$ d'une valeur x à un nombre flou est une valeur comprise entre 0 et 1, plus x est proche de l'intervalle, plus $\mu(x)$ sera élevé. La notion de proximité se définit à partir du support $[a, d]$ du nombre flou décrit par un trapèze $[a, b, c, d]$, où $[b, c]$ en est le noyau, i.e. l'intervalle initial. Etendre un intervalle $[b, c]$ en un nombre flou $[a, b, c, d]$ consiste ainsi à définir les valeurs du support $[a, b]$. La construction des nombres flous proposée par NAVIGALA intègre à la fois la distance aux intervalles voisins, ainsi que la distribution des valeurs dans l'intervalle :

$$\begin{aligned} a &= b - \theta \times \ min(d_{v^-}, d(g, c)) \\ d &= c + \theta \times \ min(d_{v^+}, d(b, g)) \end{aligned}$$

où d_{v^-} (resp. d_{v^+}) est la distance avec l'intervalle précédent (resp. suivant) issu de la même caractéristique ; g est le centre de gravité des valeurs dans l'intervalle initial ; θ est le paramètre flou.

Lorsque le centre de gravité de la distribution des valeurs est plus proche d'une des deux bornes de l'intervalle, alors le support sera plus grand du côté de cette borne. Les distances d_{v^-} et d_{v^+} sont nécessairement positives car les intervalles obtenus après discrétisation sont disjoints. Par ailleurs, chaque intervalle possède au moins un intervalle voisin car les primitives dont les valeurs n'ont pas été séparées n'apparaissent pas dans la table. Dans le cas où il n'existe qu'un seul intervalle voisin, la distance avec cet intervalle est dupliquée, et le nombre flou obtenu est alors symétrique.

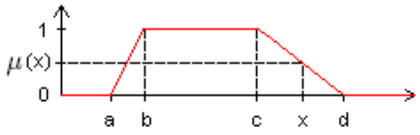


FIGURE 6.3 – Exemple de nombre flou

Apprentissage

L'apprentissage est réalisé sur une partie seulement des objets formant ainsi la *base d'apprentissage*. Il s'agit de construire le treillis des concepts de la table binaire $(O, I, (\alpha, \beta))$ des objets de la base de test. L'indication de classe est ensuite utilisée pour labéliser des concepts terminaux.

Le treillis se définit par un ensemble de concepts $\mathcal{C}$ muni d'une relation binaire $\trianglelefteq$ entre concepts (cf Définition 11 de la partie I). Rappelons que chaque concept $(A, B) \in \mathcal{C}$ est composé d'un sous-ensemble d'objets $A \in O$ et d'attributs $B \in I$ tels que $B = \alpha(A)$ et $A = \beta(B)$. La relation $\trianglelefteq$ est une relation de subsomption/généralisation définie, pour deux concepts (A_1, B_1) et (A_2, B_2), par

$$(A_1, B_1) \trianglelefteq (A_2, B_2) \iff B_1 \subseteq B_2 \iff A_1 \supseteq A_2 \tag{6.11}$$

Lors de l'étape de discrétisation, les caractéristiques dont les valeurs ne se partitionnent qu'en un seul intervalle ne sont pas considérées. Par conséquent, il ne peut exister d'intervalle associé à tous les objets, et l'élément minimal du treillis est alors égal à $\bot = (O, \alpha(O)) = (O, \emptyset)$. Par ailleurs, les intervalles construits étant disjoints, il ne peut exister d'objet associé à tous les intervalles, et le concept maximal du treillis est alors égal à $\top = (\beta(I), I) = (\emptyset, I)$.

L'algorithme de génération du treillis utilisé par NAVIGALA est une adaptation de l'algorithme de Bordat [8] qui calcule directement le diagramme de Hasse du treillis, et plus précisément la relation de couverture entre les concepts, utilisée lors de la phase de classification pour naviguer dans le treillis. Le principe de cet algorithme est simple : il s'agit de générer récursivement les successeurs d'un concept dans le diagramme de Hasse, à partir du concept minimal $\bot$. Le calcul des successeurs immédiats d'un concept (A, B) s'effectue sur la base du théorème de Bordat [86] qui établit qu'ils sont en bijection avec les sous-ensembles minimaux par inclusion d'une certaine famille définie sur $I \setminus B$ (cf Algorithme 7 de la partie I).

Une fois le treillis généré, NAVIGALA propose de sélectionner les concepts vérifiant un critère d'homogénéité défini par le même seuil que celui utilisé pour définir le critère d'arrêt de la discrétisation. Les concepts ainsi sélectionnés deviennent des *concepts terminaux* pour la classification par navigation. Plus précisément, pour chaque concept, la classe majoritaire des objets qu'il contient ainsi que le taux d'objets lui appartenant sont calculés. Lorsqu'un concept possède suffisamment d'objets d'une même classe, i.e. lorsqu'il est suffisamment pur selon le *seuil de pureté* fixé, il est alors labélisé par cette classe, et devient un *concept terminal*. Un seuil fixé à 0 indique qu'un concept sera terminal lorsque tous les objets qu'il contient appartiennent à la même classe. Lorsqu'un concept est terminal, tous ses successeurs dans le treillis le sont également, à l'exception du concept maximal $\top = (\emptyset, I)$ car il ne contient pas d'objets. Il est nécessaire que tous les co-atomes du treillis - les prédécesseurs immédiats de l'élément maximal - soient des concepts terminaux, ce qui est garanti lorsque le seuil de pureté utilisé est supérieur

ou égal à celui utilisé pour définir le critère d'arrêt de la discrétisation. C'est pourquoi la méthode NAVIGALA utilise le même seuil de pureté que celui défini pour le critère d'arrêt. Cependant, d'autres critères sont envisageables.

L'algorithme de construction du treillis de Galois utilisé en permet une génération à la demande. En effet, la reconnaissance par navigation nécessite l'exploration d'une partie seulement du treillis. C'est pourquoi les successeurs d'un concept (A, B) peuvent être générés seulement lorsqu'ils sont nécessaires à la navigation dans le treillis durant la phase de classification. La génération à la demande permet ainsi d'éviter la construction complète du treillis, dont la taille est exponentielle dans le pire des cas, et ainsi de réduire considérablement le temps de calcul de l'étape d'apprentissage.

Classification

La classification est réalisée sur une partie seulement des objets formant ainsi la *base de validation*. Il s'agit de classer chaque objet requête x de cette base, décrit par son vecteur de caractéristiques normalisé $(f_1, \dots f_n)$.

la classification s'effectue par navigation dans le treillis, à partir du concept minimal $\perp = (O, \emptyset)$, jusqu'à ce qu'un concept terminal soit atteint. La classe labélisant ce concept lui est alors affectée. Intuitivement, au cours de la progression dans le treillis, la description de l'objet requête est affinée, jusqu'à ce qu'elle soit jugée suffisamment proche d'un ensemble d'objets appartenant à un même concept terminal. A chaque étape de navigation, les intervalles des concepts successeurs du concept courant sont considérés, et le concept contenant les intervalles flous les plus proches sont sélectionnés, selon une mesure de distance et un critère de choix. Lorsque le concept terminal contient des objets de plusieurs classes - dépendant du seuil de pureté utilisé pour identifier les concepts terminaux - il est alors possible de retourner l'ensemble de ces classes triées selon le taux d'appartenance des objets du concept à chacune des classes, la classe majoritaire étant alors placée en première position.

Plus formellement, une étape élémentaire de navigation se définie pour un concept courant (A, B) - initialement $\perp = (O, \emptyset)$ - où B représente l'ensemble des attributs déjà validés - i.e. les intervalles flous - et A l'ensemble des objets possédant tous les attributs de B. Par conséquent, les classes des objets de A sont candidates à la classification de x. Le nouveau concept courant est alors sélectionné parmi l'ensemble des successeurs immédiats $(A_1, B_1), \dots, (A_m, B_m)$ de (A, B) dans le treillis, en validant un ou plusieurs attributs.

Par définition de la relation successeur du treillis, chaque ensemble d'attributs B_i contient les attributs de B car $B \subseteq B_i$. Sachant que l'ensemble des attributs de B a déjà été validé, il est possible d'identifier l'ensemble $\tilde{B}_i = B_i \setminus B$ des attributs candidats à la validation pour chaque concept (A_i, B_i). Les attributs candidats à la validation pour l'ensemble des concepts successeurs forment ainsi une famille $\tilde{B}$:

$$\tilde{B} = \{\tilde{B}_i \ : \ (A_i, B_i) \text{ successeur de } (A, B)\} \tag{6.12}$$

Cette famille forme alors une partition car les parties qu'elle contient sont disjointes. En effet, si deux parties $\tilde{B}_i$ et $\tilde{B}_j$ ne sont pas disjointes, alors $B_i \cap B_j \neq B$. La propriété de stabilité par intersection d'un treillis (cf Section 2.3) permet d'en déduire qu'il existe un concept à la fois successeur de (A, B) et prédécesseur de (A_i, B_i) et (A_j, B_j). Ce qui contredit le fait que (A_i, B_i) et (A_j, B_j) soient successeurs de (A, B) dans le diagramme de Hasse du treillis.

L'étape élémentaire de navigation considère parmi la famille d'attributs candidats $\tilde{B}$ l'ensemble d'attributs $\tilde{B}_i$ qui se rapproche au mieux du vecteur de caractéristiques de l'objet requête x. Il s'agit pour cela de définir un *critère de choix* qui repose sur une *mesure de distance* $d(x, v)$ entre le nombre flou v dont le noyau est un intervalle construit par discrétisation d'une caractéristique k, et la $k^{\text{ème}}$ caractéristique de l'objet requête x :

$$d(x, v) = d(f_i, v) = 1 - \mu(f_i, v) \tag{6.13}$$

Cette mesure intègre ainsi la distribution des valeurs de l'intervalle noyau du nombre flou, ce qui lui confère une meilleure robustesse.

Plusieurs critères du choix d'un ensemble $\tilde{B}_i$ parmi $\tilde{B}$ sont envisageables :

1. Choisir l'indice i tel que la somme des distances entre x et chaque intervalle v de $\tilde{B}_i$ soit minimal :
$$i = Argmin_{i=1\ldots m}(\Sigma_{v \in \tilde{B}_i} d(x, v)) \tag{6.14}$$
2. Choisir i tel que l'ensemble $\tilde{B}_i$ contient le nombre maximal d'intervalles parmi les k plus proches intervalles de x :
$$i = Argmax_{i=1\ldots m} |\tilde{B}_i \bigcap \{v^{(1)}, \ldots, v^{(q)}\}| \tag{6.15}$$
où les $v^{(j)}$ correspondent aux intervalles de $\tilde{B}$, triés par ordre décroissant de distances $d(x, v^{(j)})$ et q est un paramètre à fixer.
3. Choisir i tel que l'ensemble $\tilde{B}_i$ contient le nombre maximal d'intervalles situés à une distance de x inférieure à un seuil c donné :
$$i = Argmax_{i=1\ldots m} |\{v \in \tilde{B}_i \ : \ d(x, v) \leq c\}| \tag{6.16}$$

Le premier critère, défini de manière globale pour chaque $\tilde{B}_i$, possède l'inconvénient de noyer le bruit. Le second critère de choix suit le principe des k plus proches voisins, alors que le troisième critère en est un cas particulier. Ces critères étant locaux pour chaque $i = 1 \ldots n$, on peut en imaginer des plus sophistiqués par combinaison. Le critère utilisé par NAVIGALA est ainsi une combinaison de ces trois critères :

- Appliquer le troisième critère avec $c = 0$, ce qui correspond à considérer l'appartenance directe au support de chaque intervalle $v \in \tilde{B}_i$.
- Ensuite, en cas d'ambigüité, appliquer le troisième critère avec $0 < c < 1$, ce qui correspond à étendre progressivement la distance considérée, jusqu'à atteindre le support du nombre flou.

– En cas d'ambigüité, appliquer ensuite le premier critère.

Enfin, le mécanisme de *génération à la demande* propose de générer et éventuellement labéliser les successeurs d'un concept (A, B) non pas en phase d'apprentissage, mais en phase de classification, lorsque (A, B) est le concept courant. Ce mécanisme induit un coût de classification supplémentaire, mais permet de réduire considérablement le coût de la phase d'apprentissage.

Conclusion

Cette partié dédiée à la description de la méthode NAVIGALA, reprend les grandes étapes usuelles d'un processus de reconnaissance, à savoir la préparation des données, l'apprentissage et la classification :

Préparation des données. Lors de cette première étape, les données sont tout d'abord normalisées, puis discrétisées selon un *critère de coupe* et un critère d'arrêt défini par un *seuil de pureté*, en mode *global*, *local* ou encore *local linéairement*. Finalement, la binarisation permet d'obtenir une table binaire, appelée contexte en AFC, sans paramètres supplémentaires.

Apprentissage. Le treillis des concepts est généré à partir du contexte, aucun paramétrage n'est nécessaire. Ensuite, la sélection des concepts terminaux du treillis reprend le critère de pureté utilisé pour discrétiser les données. La possibilité d'une génération à la demande permet de réduire le temps nécessaire à la génération du treillis en apprentissage, tout en garantissant les mêmes taux de reconnaissance.

Classification. La classification d'un symbole à reconnaître s'effectue par une navigation dans le treillis généré, au moyen d'une mesure de distance paramétrable selon un *taux de flou*.

L'utilisation du classifieur nécessite la prise en compte de paramètres pour seulement deux des trois étapes, à savoir la discrétisation et la classification. La méthode NAVIGALA a été implémentée dans un logiciel du même nom dédié à la reconnaissance d'images de symbole, avec une première phase d'extraction des données à partir des images paramétrables par le choix d'une *signature* à extraire de l'image. Par conséquent, des expérimentations sont nécessaires pour adapter le calibrage de ces différents paramètres en fonction des données initiales. Proposée récemment, la discrétisation locale n'est pas implémentée dans ce logiciel.

Plusieurs extensions de la méthode NAVIGALA sont envisageables. Citons la possibilité d'extension au cas non supervisé qui nécessiterait une étude de critères d'arrêt de la discrétisation, et de sélection des concepts terminaux non supervisés. L'intégration d'attributs symboliques, possible lors de la phase de préparation des données, permet d'étendre la méthode NAVIGALA pour tout type de données initiales. Une adaptation de la mesure de distance à ces attributs symboliques serait néanmoins nécessaire.

Par ailleurs, l'exploitation conjointe de la génération à la demande et de la discrétisation en mode local permet d'intégrer dynamiquement de nouveaux attributs au processus de classification, et ainsi d'envisager la mise en place de mécanismes de bouclage de pertinence, et d'adaptativité de la méthode. En effet, lorsque les attributs candidats pour deux concepts successeurs C_1 et C_2 sont trop proches, il est alors envisageable d'utiliser de nouveaux attributs pour séparer aux mieux les objets de ces concepts. Ces nouveaux attributs peuvent alors être intégrés dynamiquement à la table initiale puis discrétisés en mode local, et ainsi être pris en compte dans le calcul des successeurs immédiats. Une autre possibilité serait de construire le treillis pour ces seuls nouveaux attributs, dans le but de choisir entre C_1 et C_2. Un mécanisme incrémental de génération pourrait alors être utilisé.

La méthode NAVIGALA utilise le treillis dans sa globalité pour une classification par navigation, et se distingue ainsi de la stratégie par sélection commune aux méthodes issues de l'AFC présentées dans la section 6.2 - sélection de concepts, de règles ou encore de prototypes - qui cherchent à regrouper des attributs fortement dépendants pour une même classe d'objets. La stratégie mise en oeuvre induit une différence de comportement vis-à-vis des données, et en particulier lorsque des classes sont faiblement représentées, lorsque les données sont bruitées, ou encore lorsque le nombre de classes est élevé.

Classes faiblement représentées. Dans les méthodes orientées sélection, le fait de limiter l'apprentissage aux objets les plus représentés, implique que certaines classes d'objets faiblement représentées peuvent être exclues de l'apprentissage. Ainsi avec la méthode Legal, un sous-ensemble d'objets d'apprentissage ayant des attributs très similaires, ou des objets d'apprentissage isolés, vérifieront très peu de concepts pertinents et ne pourront donc pas être reconnus. A l'inverse, dans NAVIGALA, l'ensemble des objets est conservé, sans favoriser les plus représentés, ce qui permet d'être plus exhaustif. En l'état actuel des choses, la détection de données d'apprentissage aberrantes n'est pas possible. Elles sont intégrées dans le treillis, alors qu'en sélection, les données aberrantes à faible occurrence sont bien supprimées.

Robustesse au bruit. La navigation permet d'éviter dans une certaine mesure, l'influence d'un bruit porté sur plusieurs attributs. En effet, les attributs sont validés les uns à la suite des autres et non par une moyenne, comme dans les méthodes orientées sélection. De plus, l'ordre de validation des attributs est modifiable en fonction de leur robustesse au bruit. Les attributs les plus représentés sur les objets sont proposés au début du parcours du treillis et la fréquence diminue lors de la progression dans le treillis. Enfin, la mesure de distance floue permet d'assouplir les bornes des intervalles caractérisant les attributs et d'absorber les perturbations des valeurs engendrées par le bruit. La robustesse au bruit reste un problème pour les méthodes par sélection. Alors que LEGAL résiste bien au bruit grâce à un critère de validité et de quasi-cohérence, le choix des seuils pour ces critères reste

difficile [160].

Nombre élevé de classes. Certains algorithmes de sélection ne sont pas prévus pour gérer un grand nombre de classes. Ainsi, CIBLe a des difficultés à caractériser des données contenant un grand nombre de classes surtout lorsqu'elles sont complexes [163]. En navigation, il est possible d'effectuer la classification en plusieurs phases et en utilisant différents types de signatures par intégration dynamique de nouveaux attributs, de manière itérative ou non.

6.3.2 Treillis dichotomiques

Dans le domaine de la recherche, il est essentiel de créer des ponts entre les différents outils qui peuvent être issus de divers domaines. Ces liens entre les domaines permettent de ne pas redécouvrir les mêmes résultats sous des formalismes différents. Concernant le treillis de Galois, le lien existant entre les treillis et les algorithmes génétiques [164] a été identifié dès 1987. Des points communs avec les rough sets [167] et les réseaux de neurones [166] ont également été observés. Plus récemment, il a été montré qu'un sup-demi treillis pouvait être assimilable à l'architecture d'un réseau de neurones multi-couches [160], particularité exploitée pour proposer une nouvelle approche de génération de leur architecture [186].

Ces liens sont importants car ils permettent de mieux situer les outils les uns par rapport aux autres et d'engager une réflexion plus poussée sur leurs forces et leurs faiblesses. Ainsi les liens que nous avons pu identifier entre les structures arborescentes et les treillis obtenus par la méthode Navigala sont une contribution en ce sens, qui permettent d'envisager des structures hybrides de classification intermédiaires entre arbre et treillis, dans le but à la fois d'améliorer les performances de l'arbre et de limiter la trop grande taille du treillis, tout en garantissant la lisibilité de ces structures.

Treillis dichotomiques

Les treillis générés par la méthode Navigala consituent une classe de treillis aux propriétés particulières que nous avons appelés les *treillis dichotomiques* :

Définition 15 *Un treillis est dit* dichotomique *lorsqu'il obtenu à partir d'une table de données numériques discrétisées en intervalles disjoints.*

On montre facilement que les treillis générés par Navigala possèdent deux propriétés structurelles fortes, introduites dans la section 2.2 de la partie I, à savoir la $\vee$-complémentarité et la co-atomisticité :

Co-atomisticité : Rappelons qu'un treillis est co-atomistique lorsque l'ensemble de ses inf-irréductibles - i.e. les concepts maximaux contenant les objets - sont des co-atomes - i.e. des concepts couvrant l'élément maximal. Dans la méthode Navigala, chaque objet est initialement décrit par un vecteur de caractéristiques

de même longueur. Après discrétisation, il est ensuite décrit par un même nombre d'intervalles, un intervalle pour chaque caractéristique. Par conséquent, les concepts maximaux contenant les objets - les concepts inf-irréductibles - ne peuvent être comparables entre eux dans le treillis, et sont donc nécessairement des co-atomes.

$\vee$-complémentarité : Rappelons qu'unn treillis est $\vee$-complémenté lorsque pour tout concept (A, B) du treillis, il existe toujours un concept complémentaire (A', B') tel que :

$$(A, B) \vee (A', B') = (A \cap A', \alpha(A \cap A')) = (\emptyset, I) = \top \tag{6.17}$$

Un résultat de la théorie des treillis établit que tout treillis co-atomistique est $\vee$-complémenté. Les treillis dichotomiques sont donc $\vee$-complémentés. En effet, les intervalles créés lors de la discrétisation d'une même caractéristique sont disjoints, donc mutuellement exclusifs. Ainsi, pour chaque intervalle x, si on considère l'ensemble $\overline{X}$ des intervalles distincts issus de la même caractéristique, $(x \notin \overline{X})$ alors les attributs de $\{x\} \cup \overline{X}$ sont mutuellement exclusifs, ne pouvant partager des objets communs. Par conséquent, pour tout concept (A, B) du treillis contenant un attribut x, et pour tout concept (A', B') contenant des attributs de $\overline{X}$, alors ces deux concepts ne peuvent partager des objets communs, $A \cap A' = \emptyset$. Ils sont donc $\vee$-complémentaires.

A l'inverse, tout treillis co-atomistique n'est pas toujours un treillis dichotomique. Considérons, à titre de contre exemple, les treillis booléens. Il s'agit de treillis qui sont co-atomistiques, donc $\vee$-complémentés. Cependant, ils ne sont pas dichotomiques. En effet, il n'est pas possible d'identifier, via leur table, un ensemble d'attributs complémentaires $\overline{X}$ pour chaque attribut x.

Structures arborescentes de classification

Les arbres de décision et les méthodes de classification hiérarchique sont deux approches classiques manipulant une structure arborescente, et dont nous rappelons brièvement les particularités :

Arbre de décision. Depuis les années 1960-1970, l'arbre de décision a fait l'objet de nombreux travaux de recherche [173, 174] où il peut se définir dans un contexte supervisé ou non. Il s'agit d'une structure reconnue pour sa lisibilité, ainsi que pour sa capacité à sélectionner des variables discriminantes.

Construit à partir de la racine, l'arbre de décision nécessite un critère de segmentation pour sélectionner - voire discrétiser si nécessaire - à chaque étape un attribut de la table de manière à partitionner les objets en sous-ensembles distincts formant ainsi les noeuds fils. Le processus est ensuite réitéré sur chacun des sous-ensembles, jusqu'à satisfaire un critère d'arrêt, généralement une mesure de la pureté des feuilles qui intègre l'information de classe des objets qui la composent. L'arbre est utilisé comme une structure de navigation à partir de sa racine,

jusqu'à ce qu'une feuille soit atteinte, elle porte une indication de classe qui est le résultat de la classification. A chaque noeud de l'arbre correspond une étape élémentaire de classification qui propose un ensemble d'attributs pour le test, un attribut étant proposé pour chacun des fils.

Les méthodes de génération de l'arbre de décision les plus connues sont ID3 [171], C4.5 [170], CART [129] et Chaid [147]. Ces méthodes de construction d'un arbre se distinguent de par les critères de discrétisation utilisés (global ou local, supervisé ou non supervisé, critère de coupe, ...) mais aussi de par la stratégie utilisée dans une phase d'élagage de l'arbre, phase de test intermédiaire entre apprentissage et classification qui permet d'améliorer l'arbre en supprimant certaines branches.

Classification hiérarchique. Les méthodes de classification hiérarchique sont des méthodes classiques de classification non supervisée - segmentation - où il s'agit de réaliser des regroupements au sein d'un ensemble d'objets. Ces méthodes visent à construire un *dendogramme*, structure arborescente valuée qui illustre l'arrangement de groupes d'objets générés par un regroupement hiérarchique. La valuation du dendogramme permet de définir une segmentation comme la troncature de l'arbre à un niveau donné, troncature qui forme alors une partition de l'ensemble des objets en différents clusters. Il existe plusieurs partitions de plus en plus fines : la première indique deux clusters (troncature après la racine) ; la dernière indique un objet par cluster (troncature avant les feuilles).

La hiérarchie peut être générée de façon ascendante ou descendante. La classification hiérarchique ascendante procède par fusions successives de regroupements - ou clusters - déjà existants, les objets sont considérés comme des clusters singletons. A chaque étape, les deux clusters les plus proches sont fusionnés, où la proximité dépend d'une mesure de distance entre clusters. Le processus s'arrête lorsque le cluster fusionné contient tous les objets, ce qui correspond à la racine. On trouve plusieurs méthodes de classification hiérarchique ascendante, qui se distinguent de par la mesure de distance utilisée.

A l'inverse, la classification hiérarchique descendante considère initialement l'ensemble des objets comme un seul cluster, puis opère par scissions successives jusqu'à obtenir des clusters singletons correspondant aux feuilles de l'arbre. La scission est réalisée de façon à créer deux clusters bien séparés, de distance maximale. Le principal inconvénient de cette méthode réside dans le coût de l'opération de scission car il s'agit de tester toutes les scissions possibles. Cette difficulté est habituellement contournée en considérant les scissions sur la base des valeurs prises par les objets sur une seule variable.

La technique classification hiérarchique descendante, également connue sous la dénomination d'arbre de segmentation, est très proche de l'arbre de décision. L'arbre de décision est utilisé dans le cas supervisé, ce qui lui permet d'intégrer les modalités d'une variable de classe pour une scission plus efficace.

Liens entre treillis dichotomiques et structures arborescentes

La méthode NAVIGALA propose un *mécanisme de navigation* similaire à celui de l'arbre de décision. La navigation dans ces deux structures se distingue cependant de par le fait que l'arbre propose à chaque étape de classification un ensemble d'attributs à tester, alors qu'il s'agit d'une famille d'attributs - plus précisément une partition - pour NAVIGALA. Par conséquent, la navigation proposée au sein du treillis est plus générique. La navigation au sein d'un arbre en est une restriction, avec une famille composée d'ensembles singletons, ce qui simplifie l'expression du critère de choix.

En conséquence, il est possible d'associer un unique concept à chaque noeud n de l'arbre de décision via la connexion de Galois (α, β) de la table binarisée de ses attributs discrets. Ce concept se définit par :

$$(\beta(B_n), \alpha(\beta(B_n)). \tag{6.18}$$

Il s'agit du plus petit concept contenant l'ensemble des attributs B_n proposés depuis la racine jusqu'au noeud n. Par ailleurs, pour tout noeud n' du sous-arbre issu de n, l'inclusion $B_n \subseteq B_{n'}$ clairement vérifiée implique que le concept associé à n' est successeur du concept associé à n selon la relation $\trianglelefteq$ entre concepts. Ce qui permet de déduire que tout arbre est inclus dans le treillis, lorsqu'ils sont définis pour un même ensemble d'attributs discrets.

Les arborescences hiérarchiques sont quant à elles centrées objets et non attributs. Une hiérarchie est une famille $\mathcal{F}$ de parties d'objets, qui contient l'ensemble de tous les objets, mais également les singletons, et qui vérifie la propriété suivante pour deux parties distinctes F et F' de la famille :

$$F \cap F' = \emptyset \text{ ou } F \subseteq F' \text{ ou } F' \subseteq F \tag{6.19}$$

La famille $\mathcal{F}$ ordonnée par inclusion forme nécessairement un arbre. De plus, la famille $\mathcal{F} \cup \emptyset$ vérifie la propriété de stabilité par intersection, propriété caractéristique d'une famille de Moore, ou encore d'un treillis des fermés introduite dans la section 2.3. Par conséquent, toute arborescence hiérarchique est inclue dans le treillis des fermés définis sur le même ensemble d'objets.

Ces deux constations permettent facilement d'établir qu'une structure arborescente définie à partir de données discrètes est inclue dans le treillis de la table binarisée. L'inclusion de l'arbre dans le treillis est vérifiée dans le cas général. Ce lien d'inclusion se trouve renforcé par un lien de fusion vérifié pour les treillis dichotomiques :

Lien d'inclusion. Toute structure arborescente est inclue dans le treillis.

Lien de fusion. Le treillis dichotomique est la fusion de toutes les structures arborescentes.

Le lien d'inclusion est une conséquence immédiate de la propriété de fermeture d'un treillis. La propriété de fusion se déduit quant à elle de la $\vee$-complémentarité.

Elle est par conséquent propre aux treillis dichotomiques et renforce ainsi les liens structurels entre treillis dichotomiques et arbres de classification. Nous en donnons une preuve constructive dans [18] en montrant que tout sous-arbre maximal d'un treillis dichotomique est un arbre qui sépare les objets de classes différentes.

De cette preuve nous avons déduit un algorithme d'extraction d'un arbre de décision à partir d'un treillis dichotomique [18]. Il s'agit d'un algorithme qui prend en paramètre l'arité de l'arbre à générer, un critère de sélection d'un noeud parmi un ensemble de concepts ainsi qu'un critère d'arrêt précisant s'il s'agit ou non d'une feuille (souvent une mesure pureté du concept). Initialement appelé pour le concept minimal du treillis, cet algorithme sélectionne à chaque étape un concept successeur (A, B), puis un ou plusieurs concepts complémentaires de (A, B) selon les paramètres de sélection et d'arité. Les concepts ainsi sélectionnés seront des fils du concept courant dans l'arbre, et le processus est réitéré jusqu'à ce que les concepts sélectionnés vérifient le critère d'arrêt. L'arbre ainsi généré est défini à partir des mêmes attributs binaires que le treillis. Différents critères de sélection sont envisageables permettant d'intégrer à la fois des informations statistiques (concept le plus hétérogène, ...), mais aussi structurelles (concept dont le sous-arbre est de hauteur /largeur minimale/maximale).

Les liens structurels similaires entre treillis et structures arborescentes ont été explorés sous différentes formes. Citons quelques travaux menés en ce sens :

1. La propriété de $\vee$-complémentarité à été exploitée pour introduire la notion de table dichotomique dans [58]. Une *table dichotomique* y est définie par l'existence d'un unique attribut complémentaire $\overline{x}$ pour tout attribut binaire x. La propriété d'unicité étant ici ajoutée à celle de complémentarité, ceci signifie que tout objet possède soit l'attribut x, soit son complémentaire $\overline{x}$. Il est possible de transformer une table binaire quelconque en une table dichotomique en lui ajoutant tous les attributs complémentaires manquants. Le nombre d'attributs dans la table peut alors doubler et le nombre de concepts du treillis peut croître de façon exponentielle. Il apparaît clairement que tout treillis issu d'une table dichotomique est un treillis dichotomique, alors que l'inverse n'est pas toujours vrai. Plus précisément, ces treillis sont $\vee$-complémentés, et vérifient en plus la propriété d'unicité du concept complémentaire.
2. Une comparaison structurelle [151, 152] entre le treillis issu d'une table dichotomique et tout arbre de décision issu de cette même table a permis d'établir les résultats suivants :
   - Toute branche maximale de l'arbre correspond à une chaîne maximale du treillis, où une branche maximale est une section de l'arbre partant de la racine allant jusqu'à une feuille et contenant tous les noeuds intermédiaires.
   - Toute chaîne maximale du treillis correspond à une branche maximale de l'arbre, où une chaine maximale est une chaîne du diagramme de Hasse du treillis partant du concept minimal allant jusqu'au concept maximal et conte-

nant tous les concepts intermédiaires.

3. Les travaux décrit dans [131] s'intéressent à l'étude théorique de structures de classification utiles en analyse des données, c'est à dire ayant des propriétés plus générales que les hiérarchies mais tout de même représentables globalement avec un nombre polynomial de classes. Ces travaux ont mis en avant la classe des *treillis démentalables* [132] qui forment une généralisation aux treillis des arbres de classification.
4. Dans une étude récente [126], un arbre de hauteur maximale est extrait à partir d'un treillis quelconque. Ses taux de reconnaissance améliorent ceux des arbres de décision générés par CART ou C4.5 sur plusieurs bases, offrant ainsi des perspectives intéressantes.

Perspectives

Une application directe du lien de fusion entre un treillis dichotomique et les structures arborescentes issues des mêmes données, conséquence directe de la $\vee$-complémentarité des treillis dichotomiques, serait d'extraire du treillis des structures arborescentes qu'il s'agirait ensuite de fusionner pour créer une structure hybride de classification intermédiaire entre arbre et treillis, dans le but à la fois d'améliorer les performances de l'arbre et de limiter la trop grande taille du treillis, tout en garantissant la lisibilité de ces structures.

Une autre approche serait de ne conserver dans le treillis que les concepts suffisamment pertinents pour la navigation. Dans des travaux récents [148], une étude comparative entre quelques mesures de pertinence proposées pour une meilleure robustesse au bruit laisse envisager des perspectives intéressantes pour la stratégie de classification par navigation. En particulier, la mesure de stabilité d'un concept [153], qui mesure la proportion de parties des objets d'un concept dont la fermeture correspond à ce même concept, permet d'identifier si un concept dépend ou non d'objets particuliers. La structure composée des concepts les plus stables, structure hybride entre arbre et treillis car les concepts stables peuvent se situer n'importe où dans le treillis, laisse également envisager une amélioration de la stratégie de classification par navigation.

A l'inverse, les critères plus classiques de sélection de concepts proposés dans le but de réduire la taille du treillis s'apparentent le plus souvent à un mécanisme d'élagage dans le treillis. Ainsi, l'extraction de règles d'association utilise la mesure de fréquence pour réduire le treillis à sa partie inférieure appelée Iceberg de Galois. Les méthodes de classification issues de l'AFC utilisent quant à elles une mesure supervisée d'homogénéité pour sélectionner un sous-ensemble de concepts situé dans sa partie supérieure.

Les propriétés des treillis dichotomiques - conséquence directe de la discrétisation en intervalles disjoints proposée par NAVIGALA - permettent d'envisager une algorithmique adaptée à cette classe de treillis. Par exemple, la propriété de co-atomisticité permet d'améliorer le coût de la discrétisation locale que nous avons proposée. En effet, à

chaque étape de discrétisation, des concepts particuliers à discriminer sont générés, il s'agit des inf-irréductibles du treillis, à l'aide d'un algorithme de génération des inf-irréductibles dont la complexité est élevée (cf section 4.3 de la partie I). Lorsque le treillis est co-atomistique, alors tous les inf-irréductibles sont les co-atomes du treillis, et leur génération consiste à calculer les seuls co-atomes à l'aide d'un algorithme plus efficace, et ainsi d'en améliorer le temps de calcul. En effet, les co-atomes étant les concepts prédécesseurs immédiats du concept maximal $\top$, il suffit pour les obtenir de calculer les successeurs immédiats du concept minimal $\bot$ du treillis de la table inversée. L'algorithme de discrétisation locale proposé laisse également envisager un mécanisme de génération incrémentale spécifique aux treillis dichotomiques. Des travaux en ce sens sont en cours.

6.4 Expérimentations

Cette section a pour objectif de présenter plusieurs résultats expérimentaux. Ces expérimentations ont pour la plupart été réalisées en collaboration avec Jean-Marc Ogier et Stéphanie Guillas, dans le cadre de ses travaux de thèse [40]. Les expérimentations autour de la discrétisation locale ont quant à elles été réalisées en collaboration avec Muriel Visani et Nathalie Girard, dans le cadre de ses travaux de thèse en cours. L'expérimentation sur la génération multi-processeur d'un treillis a été réalisée en collaboration avec Antoine Mercier et Romain Bertrand, dans le cadre de son stage de master [37].

La première série d'expérimentations (Section 6.4.1) a pour but de paramétrer la méthode Navigala dans un objectif de reconnaissance de symboles. Une seconde série d'expérimentations (Section 6.4.2) vise à positionner Navigala par rapport à d'autres classifieurs usuels pour des images de symboles sur les symboles, ou encore à des méthodes issues de l'AFC pour des données numériques quelconques.

6.4.1 Paramétrage de la méthode Navigala

Un premier objectif de cette expérimentation concerne le réglage des paramètres de la méthode Navigala. Cette méthode ayant été développée dans un objectif de reconnaissance de symboles, les tests ont été réalisés sur des images de symboles segmentés de la base GREC : GREC 2003 et GREC 2005 (Graphics RECognition) [175, 176].

Symboles de la base GREC

Les images que nous considérons sont des images pré-segmentées issues de documents techniques, de plans architecturaux, ou encore de diagrammes électriques (cf Figure 6.4). Il s'agit d'images détériorées, et les origines de la détérioration des images sont variées : détérioration du papier (taches, parties effacées), des objets ou des distorsions de numérisation vectorielle dans le cadre des symboles écrits à la main.

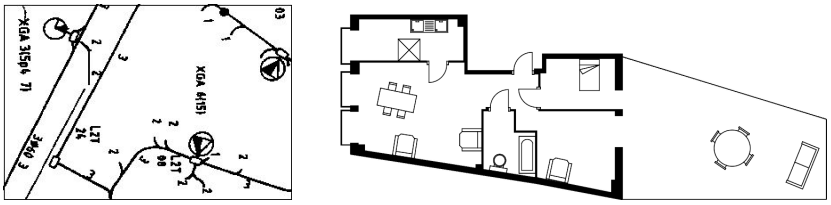


FIGURE 6.4 – Exemples de symboles

Plusieurs signatures sont extraites des images de symboles : des signatures statistiques (les invariants de Fourier-Mellin [137], la transformée de Radon [172], les moments de Zernike [184]), ainsi qu'une signature structurelle adaptée aux symboles que nous avons développée [19], décrite dans la section 7.1.2. Rappelons que les signatures statistiques décrivent la distribution spatiale des valeurs des pixels, alors que les signatures structurelles décrivent l'organisation spatiale ou topologique des relations entre des primitives issues de l'image.

Les bases GREC [175, 176] (Graphics RECognition) sont composées d'images de symboles segmentés, c'est-à-dire isolés. Elles ont été créées pour un concours de reconnaissance de symboles organisé au cours de l'atelier Graphics Recognition qui a lieu tous les 2 ans. Les bruits appliqués sur les images reproduisent les déformations qui peuvent survenir lors de l'utilisation d'un photocopieur, d'un scanner ou encore d'une imprimante. Ils sont générés par la méthode Kanungo [146]. Trois catégories de bruits sont proposées : les dégradations binaires (bruit impulsif), les distorsions vectorielles et les images réelles numérisées et photocopiées.

La base GREC 2003 contient 35139 images symboles réparties en 39 classes de 901 symboles chacune. Chaque classe est composée d'un symbole modèle ainsi que de 900 symboles bruités (10 symboles pour 9 types de dégradations). La base GREC05 contient 175 images de symboles pour 25 classes. Chaque classe est composée d'un symbole modèle, et d'un nombre variable de symboles détériorés pour 6 types de distorsions. Notons que les bruits appliqués sont nettement plus forts que dans la base GREC 2003. Des exemples de symboles modèles et dégradés sont présentés dans les figures 6.5 et 6.6.

Critère de coupe de la discrétisation

Dans cette expérimentation, nous avons choisi de tester l'adéquation des critères de coupe à notre système de reconnaissance. Les critères de coupe testés sont ceux présentés dans la section 6.2.1 de préparation des données : la distance maximale, l'entropie et le coefficient de Hotelling.

Pour évaluer ces trois critères, nous avons utilisé un jeu de données issu de la base GREC 2003. L'apprentissage est réalisé avec un symbole modèle pour chacune des 10 classes, et la classification sur 90 symboles bruités par classe. La figure 6.7 présente les taux de reconnaissance, et la figure 6.8 la taille du treillis, pour chacun des trois

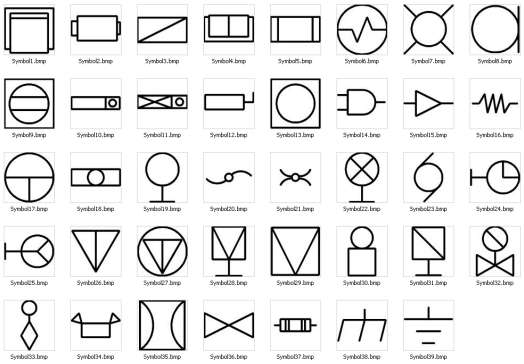


FIGURE 6.5 – Symboles non bruités de la base GREC 2003

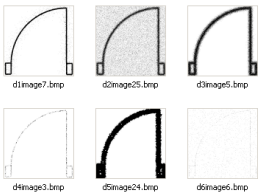
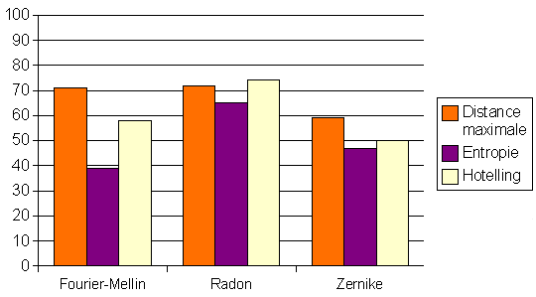


FIGURE 6.6 – Symboles bruités de la base GREC 2005

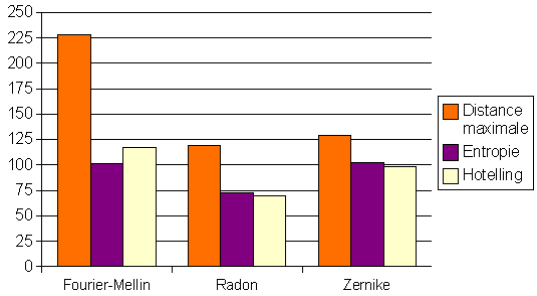
critères de coupe. Les signatures utilisées sont les signatures de Fourier-Mellin, Radon et Zernike.

FIGURE 6.7 – Taux de reconnaissance obtenus selon le critère de coupe



Ces résultats nous ont amené à sélectionner la distance maximale comme critère non supervisé et le coefficient de Hotelling comme critère supervisé pour la reconnaissance de symboles. En effet, ces deux critères de coupe obtiennent les meilleurs taux de reconnaissance quelle que soit la signature utilisée. D'autre part, le treillis généré à partir du critère de Hotelling contient en général un moins grand nombre de concepts qu'avec les autres critères de coupe. Cette propriété nous intéresse étant donné que plus la structure du treillis est réduite, plus la classification est rapide.

FIGURE 6.8 – Taille du treillis selon le critère de coupe



Comparaison des signatures

Nous avons souhaité comparer l'efficacité des signatures statistiques et structurelles. En réalité, on ne peut évaluer que leur adéquation au système de reconnaissance, étant donné qu'elles sont insérées au sein d'une chaîne de traitement, où il peut y avoir des corrélations entre les différents éléments utilisés. Nous avons effectué ces tests sur un ensemble d'apprentissage composé de 8 classes (10 symboles appris par classe). Les taux de reconnaissance obtenus sont présentés dans la figure 6.9, et les tailles de treillis correspondantes dans la figure 6.10.

FIGURE 6.9 – Taux de reconnaissance obtenus pour les différentes signatures

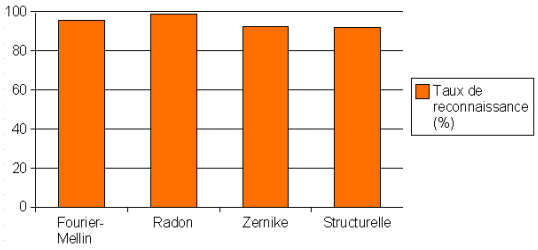
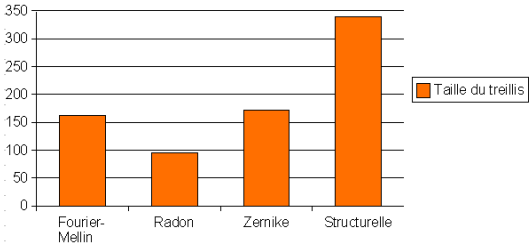


FIGURE 6.10 – Tailles des treillis obtenus pour les différentes signatures



Si l'on compare les treillis obtenus sur le même ensemble de symboles d'apprentissage mais décrits par différentes signatures, on constate que la taille du treillis est variable.

TABLE 6.6 – Apprentissage avec 25 symboles modèles - Reconnaissance de 10 symboles bruités

	Apprentissage	Classification	Nombre de concepts
Treillis entier	430,2 sec	2 sec	3185
Génération à la demande	0,5 sec	9,8 sec	282

Intuitivement, une signature efficace propose un treillis de petite taille, car les objets d'une même classe sont regroupés dans les mêmes concepts (voir dans le même concept). En effet, la signature permet de différencier les objets de classes différentes tout en permettant un regroupement des objets d'une même classe. Un treillis de petite taille signifie qu'il y a peu d'objets isolés (sinon, il y aurait plus de concepts). Ainsi, la signature de Radon offre des taux de reconnaissance intéressants et une taille de treillis relativement faible.

Génération à la demande

Nous avons également réalisé une étude comparative pour tester l'efficacité de l'algorithme de génération à la demande par rapport à la génération du treillis entier.

Le tableau 6.6 présente les durées obtenues en apprentissage et en classification ainsi que le nombre de concepts construits dans le treillis avec et sans génération à la demande. L'ensemble d'apprentissage contient 25 symboles modèles (donc 25 classes) et l'ensemble de test contient 10 symboles bruités.

Précisons tout d'abord que les taux de reconnaissance obtenus avec et sans génération à la demande sont bien identiques puisque le procédé de classification reste inchangé. D'après ces résultats, on observe qu'en apprentissage, l'algorithme de génération à la demande est très rapide étant donné que seul le concept minimal du treillis est généré. En revanche, la classification est plus longue que dans la génération du treillis entier car les concepts nécessaires sont construits au fur et à mesure de la navigation dans le treillis. Enfin, le nombre de concepts obtenus est bien moins important pour la génération à la demande.

Si les symboles à reconnaître sont très diversifiés et utilisent tous les scénarii de classification possibles, le treillis généré à la demande ne sera pas plus rapide, car presque tous les concepts du treillis seront construits pendant la phase de classification. A l'inverse, si les symboles à reconnaître ne correspondent qu'à un seul scénario de classification, le gain de temps offert par la génération à la demande sera très important.

Génération multi-processeur du treillis

Dans cette expérimentation, nous avons voulu tester la possibilité d'une génération mutli-processeur d'un treillis par un calculateur multi-coeur. L'algorithme utilisé par

TABLE 6.7 – Comparaison entre une génération du treillis avec un ou plusieurs processeurs

	Signature		Taille treillis		Temps génération itérative	
	type	longueur	nb concepts	nb arcs	mono-processeur	multi-processeur
Petit	Radon	12	189	562	1 sec.	1,5 sec.
	Zernique	12	519	1960	1,5 sec.	3,5 sec.
Moyen	Radon	34	8225	35 027	6 mn	1,4 mn
	Zernique	34	25 985	144 034	49 mn	**3 mn**
Grand	Radon	50	14 981	63 582	24 mn	**2,15 mn**
	Zernique	50	155 057	976 777	$>$ 20h	**30 mn**

NAVIGALA est un algorithme récursif. C'est pourquoi nous avons utilisé une version itérative de l'algorithme de Bordat (cf Algorithme 10 de la section 4.4 de la première partie). La génération multi-processeur du treillis consiste à créer un sous-processus thread pour chacun des concepts à traiter à chaque étape de l'algorithme. Le thread sera chargé d'exécuter la méthode de génération des successeurs immédiats de ce concept à l'aide. Une amélioration consiste à calculer un sous-treillis de rang n (paramètre à fixer).

Deux conteneurs synchronisés sont utilisés. Le premier permet de stocker l'ensemble des noeuds à traiter, et le second contient les arcs du treillis générés. Chaque arc est décrit par un concept origine et un concept destination. Garder en mémoire l'ensemble des arcs suffit donc pour reconstruire l'intégralité d'un treillis. Un mécanisme de comparaisons de concepts par le contenu permet de gérer simplement les doublons.

Le protocole de l'expérimentation est le suivant. Le calculateur utilisé est un MacOSX SnowLeopard 10.6.3, avec un microprocesseur Intel Core2Duo 2,4Ghz, une vitesse de bus 800Mhz, et une mémoire vive de 2Go DDR2 SDRAM 667Mhz. Huit coeurs sont utilisés. Six jeux de test ont été créés, qui se distinguent selon le type et le nombre de caractéristiques de la signature extraite. Ces jeux de tests sont répartis en trois catégories selon la taille du treillis généré (petit, moyen et gros). Plus la signature contient de caractéristiques, plus la taille du treillis est grande. Les treillis sont générés en utilisant la version itérative avec et sans la création de thread à chaque étape.

La table 6.7 présente les résultats obtenus. Il apparaît clairement que nous obtenons un gain en temps conséquent en utilisant plusieurs threads lorsque le treillis possède plus de 25.000 concepts. Par ailleurs, l'utilisation de plusieurs threads a permis de générer les 155.057 concepts et les 976.777 arcs du treillis du dernier jeu de données en 30 minutes, alors que sa génération avec un seul thread a du être interrompue au bout de 20 heures de calculs.

Comparaison avec les règles

Dans cette expérimentation, nous avons choisi de comparer les dimensions de certaines structures de l'AFC obtenues à partir d'une table discrétisée, à savoir le nombre de concepts du treillis, ainsi que le nombre de règles des deux bases exactes que sont la base canonique et la base canonique directe (cf Section 3.2.2). L'expérimentation a été réalisée pour les symboles de la base GREC 2003 décrits par la signature de Radon, et discrétisés selon le critère de Hoteling. Rappelons qu'à partir de la base canonique directe, il est possible d'obtenir le base de règles d'association générique et informative (cf Section 6.2.2).

Nous avons choisi de comparer les dimensions et taux de reconnaissance de ces structures dans le cas d'une validation croisée à 5 blocs. Dans cette expérimentation, chacun des cinq blocs de la validation croisée contient 91 symboles provenant de 5 classes. La table 6.8 présente le nombre d'attributs $|S|$ de la table discrétisée, le nombre de concepts du treillis, le nombre de règles dans la base canonique Σ_{can} et le nombre de règles dans la base canonique directe Σ_{cd} pour les cinq blocs de la validation croisée. Nous donnons également les taux de reconnaissance obtenus en utilisant le treillis de Galois à titre indicatif.

TABLE 6.8 – Dimensions et taux de reconnaissance obtenus en validation croisée (5 blocs)

	Bloc 0	Bloc 1	Bloc 2	Bloc 3	Bloc 4
$\|S\|$	7	8	8	6	7
Nombre de concepts	20	42	24	25	23
$\|\Sigma_{can}\|$	33	62	32	31	32
$\|\Sigma_{cd}\|$	280	779	724	103	293
Taux de reconnaissance (%)	98,6	99,2	99,2	98,9	99,2

Dans la littérature, les bases de règles sont réputées pour être une représentation condensée du treillis de Galois. Or dans la table 6.8, on constate que le nombre de concepts du treillis est moins important que le nombre de règles de la base canonique Σ_{can}, et beaucoup moins important que le nombre de règles de la base canonique directe Σ_{cd}. Il est possible d'expliquer cette différence par le fait que les treillis générés par NAVIGALA pour la reconnaissance de symboles sont co-atomistiques - i.e. les concepts "finaux" contenant une seule classe, sont toujours couverts par le concept max - alors que cette propriété n'est pas vérifiée dans le cas général. D'autre part, les données étudiées sont très différentes. En effet, les données de la base GREC sont particulières car les objets sont décrits par un très grand nombre d'attributs (de très nombreuses signatures peuvent être calculées sur les images), alors que le nombre d'attributs est bien moindre,

comme c'est le cas habituellement sur les données issues de la fouille de données. Cette différence a certainement une influence sur les résultats observés.

Notons que lorsque le nombre de concepts augmente, la taille des bases diminue, et inversement. Dans le pire des cas, lorsque le nombre de règles est nul ($|\Sigma| = 0$), le nombre de concepts du treillis est maximal : $2^{|S|}$. Inversement, plus le nombre de règles est élevé, plus les règles risquent d'être redondantes entre elles, et plus la taille de la base diminue entraînant une augmentation du nombre de concepts.

En conclusion, le treillis des concepts semble plus approprié que les bases de règles sur les jeux de données GREC. En effet, dans ce cadre, le treillis conserve une taille inférieure à celle des bases de règles. Ce résultat peut s'expliquer par la propriété de co-atomicité.

6.4.2 Résultats expérimentaux

Comparaison avec quelques classifieurs standards

Dans cette expérimentation, nous avons comparé les résultats obtenus par NAVIGALA avec des classifieurs usuels que sont le classifieur Bayésien et la méthode SVM. Nous avons utilisé des jeux de données issus des deux bases GREC : deux jeux de données composés de 10 classes chacun (jeux cl1-10 et cl11-20) issus de la base GREC 2003, ainsi qu'un jeu de 25 classes issu de la base GREC 2005 (jeu cl1-25). Les symboles sont décrits par un vecteur de 50 caractéristiques représentant la signature de Radon. Les classifieurs sont évalués en validation croisée, avec des tailles variant pour chacun des jeux de données : 5 blocs de 182 symboles de GREC 2003 (Test1), 10 blocs de 91 symboles de la base GREC 2003 (Test2), 26 blocs de 35 symboles de la base GREC 2003 (Test3), et 5 blocs de 35 symboles de la base GREC 2005 (Test4). Les taux moyens de reconnaissance sont donnés par la Figure 6.11.

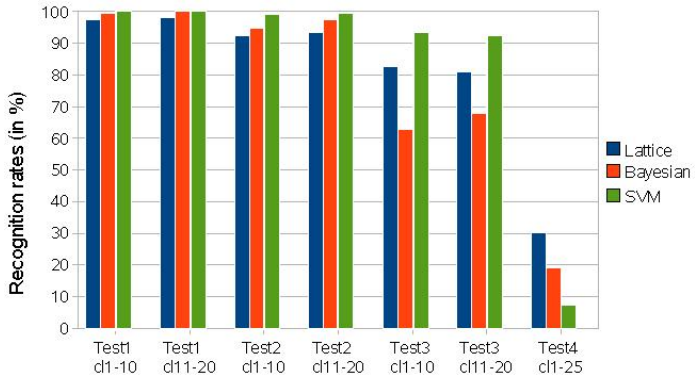


FIGURE 6.11 – Taux de reconnaissance de quatre classifieurs en validation croisée

Alors que le classifieur bayésien obtient de meilleurs résultats que NAVIGALA pour des bases faiblement bruitées de 10 classes (Test1 et test2), NAVIGALA dépasse le classi-

fieur bayésien lorsque le nombre de classes augmente (Test3) ou encore lorsque le bruit est important (Test4). Nous pouvons ainsi en déduire que, dans les expérimentations que nous en avons faite, l'approche NAVIGALA est plus robuste au bruit que le classifieur bayésien lorsque le nombre de classes augmente. NAVIGALA obtient également de meilleurs résultats que le classifieur SVM sur le Test4.

Par ailleurs, NAVIGALA réalise une sélection de caractéristiques réalisée en prétraitement de la classification pendant la phase de discrétisation, car seules les caractéristiques dont les valeurs sont partitionnées en plusieurs intervalles sont conservées. Ainsi, NAVIGALA n'utilise que entre 6 et 15 caractéristiques de la signature, alors que les autres classifieurs utilisent la signature dans sa globalité.

Comparaison avec l'arbre de décision

Pour comparer NAVIGALA et l'arbre de décision, deux expérimentations ont été réalisées, qui diffèrent selon que la discrétisation est réalisée en mode global ou local.

Dans la première expérimentation, la phase de discrétisation a été réalisée de manière commune pour que l'arbre de décision et le treillis de Galois reçoivent les mêmes données en entrée. Il s'agit d'une discrétisation globale supervisée utilisant le coefficent de Hoteling. Les deux structures sont ensuite générées à partir de la table discrétisée, l'algorithme CART [129] est utilisé pour l'arbre de décision. Nous considérons 10 classes de la base GREC 2003 (10 symboles modèles en apprentissage, et 90 symboles détériorés en classification - 10 par classe). Les symboles sont décrits par un vecteur de 50 caractéristiques représentant la signature de Radon.

Nous obtenons un taux de reconnaissance de 57% avec l'arbre de décision, alors qu'il est de 72% avec le treillis. Une des principales différences entre ces deux structures réside dans le fait qu'il existe dans le treillis de Galois plusieurs chemins qui permettent d'aboutir à une même classe alors qu'il n'y a qu'une seule possibilité dans l'arbre de décision. L'expérimentation ayant été menée sur les mêmes données discrétisées, on peut en conclure que l'existence de plusieurs chemins permet d'améliorer le taux de reconnaissance, rendant ainsi le treillis plus robuste au bruit.

Dans notre expérimentation, 9 étapes de discrétisation ont été nécessaires pour ainsi obtenir 17 intervalles discrétisés. Le treillis est composé de 70 concepts, alors que l'arbre ne contient que 18 noeuds. En effet, alors que la taille de l'arbre reste polynomiale en la taille des données, celle du treillis est beaucoup plus élevée, pouvant être exponentielle dans le pire des cas. Cependant, le nombre de concepts générés, ainsi que la complexité, sont largement réduits par une génération à la demande des concepts.

Dans la seconde expérimentation, la discrétisation est réalisée localement pour l'arbre de décision et pour le treillis de Galois, avec pour critère de coupe le critère du χ^2. L'arbre de décision est construit par l'algorithme CHAID [147] qui implémente ce critère de coupe. Pour NAVIGALA, les trois modes de discrétisation sont utilisés - discrétisation globale, locale, et linéaire locale - chacun d'eux produit un treillis différent.

L'expérimentation a été réalisée pour les symboles de la base GREC 2003 décrits par la signature de Radon, et par la signature structurelle que nous avons développée, ainsi que sur trois bases de données issues de l'UCI Repository [139] que sont Image1, Glass et Iris.

La base Image1 est composée d'images de scènes naturelles, chaque image est décrite par un vecteur de 19 attributs numériques. La base d'apprentissage est composée de 210 objets (30 objets par classe), et la base de test de 2100 objets (300 objets par classe). La base Glass est composée de 214 objets répartis en 6 classes. Chaque objet est décrit par 9 attributs numériques. Nous utilisons une validation croisée en 10 blocs. La base Iris est composée de 150 exemples répartis en 3 classes, où chaque objet est décrit par 4 attributs numériques. Nous utilisons également une validation croisée en 10 blocs.

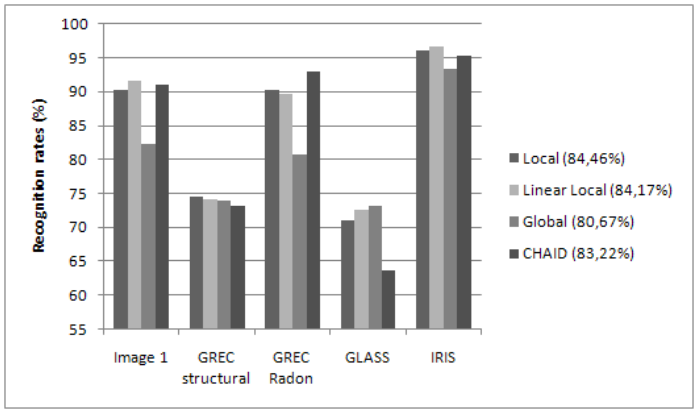


FIGURE 6.12 – Taux de reconnaissance en validation croisée

Les taux moyens de reconnaissance obtenus sont donnés par la figure 6.12. Le treillis avec une discrétisation locale améliore les taux de reconnaissance de l'arbre sur toutes les bases, à l'exception de la base Grec Radon pour laquelle les résultats sont similaires. Cependant, l'arbre de décision permet d'obtenir de meilleurs résultats qu'avec le treillis en mode global. Ces résultats confirment la meilleure robustesse au bruit du treillis par rapport à l'arbre.

Par ailleurs, bien qu'on observe peu de différence entre les trois modes de discrétisation pour les bases GrecStruct et Glass, les discrétisations locale et linéaire locale permettent d'obtenir de bien meilleurs résultats que la discrétisation globale pour les autres bases. Nous avons voulu comparer les structures générées par NAVIGALA pour chacun des trois modes de discrétisation en terme de nombre d'étapes de discrétisation, et nombre de concepts du treillis (cf tableau 6.9). Il apparaît que le nombre d'étapes de discrétisation est largement inférieur en discrétisation locale et locale linéaire qu'en discrétisation globale. Ce qui indique que la discrétisation locale permet d'atteindre une homogénéité des concepts finaux plus rapidement que la discrétisation globale, en considérant localement les concepts à discrétiser. Le nombre de concepts du treillis reste

également le plus faible pour une discrétisation locale, d'où un gain important en temps de calcul. En conclusion, sur les bases considérées, la discrétisation locale permet de réduire la taille des structures, tout en garantissant de bons taux de reconnaissance.

TABLE 6.9 – Nombre d'étapes de discrétisation et de concepts générés par NAVIGALA sur chaque base

Bases de données		Mode local	Mode linéaire local	Mode global
Grec Struct.	Nombre étapes disc.	24	24	171
	Nombre concepts	**3515**	3851	4308
Grec Radon	Nombre étapes disc.	11	10	85
	Nombre concepts	**69**	92	2192
Image1	Nombre étapes disc.	17	15	49
	Nombre concepts	**527**	649	12172
Glass	Nombre étapes disc.	24	21	72
	Nombre concepts	**1950**	2128	2074
Iris	Nombre étapes disc.	9	9	27
	Nombre concepts	**40**	42	195

En conclusion, la discrétisation locale apparaît intéressante pour la méthode NAVIGALA car elle permet à la fois d'améliorer les taux de reconnaissance de la discrétisation globale, tout en réduisant le temps de calcul nécessaire à la génération du treillis.

Comparaison avec des classifieurs issus de l'AFC

Nous avons effectué des comparaisons expérimentales sur les méthodes de classification par sélection issues de l'AFC. Les bases de données utilisées sont issues de l'UCI Repository [139] : Breast-cancer (BC), Iris (IR), Monks1, Monks2 et Monks3 (M1, M2, M3), Soybean-small (SS) et Zoo (ZO) (cf Tableau 6.10). Nous avons repris les résultats expérimentaux disponibles dans les articles décrivant les méthodes : RULEARNER, CIBLe, CLNB - CLNN, et C4.5Rules. De ce fait, nous ne sommes pas en mesure de présenter pour ces méthodes, les taux de reconnaissance de l'ensemble des bases de données. Le tableau 6.12 présente les taux d'erreur de classification obtenus en validation croisée.

En général, NAVIGALA obtient des taux d'erreurs de classification relativement proches de ceux obtenus pour les autres classifieurs sauf pour les bases Monks2 et Monks3. Ces deux bases ne correspondent pas au type de données que l'on souhaite traiter avec NAVIGALA. En effet, ces bases contiennent seulement 2 classes, cas très rare en classification de symboles. D'autre part, sur la base Monks3, 5% de bruit est appliqué sur les classes (permutation des étiquettes de classe des signatures). Ce type de bruit est

TABLE 6.10 – Bases de données du ML Repository

Bases de données	Nombre d'objets	Nombre d'attributs		Nombre de classes
		Numériques	Symboliques	
BC	699	9	0	2
IR	150	4	0	3
M2	432	0	6	2
M3	432	0	6	2
SS	47	0	35	4
ZO	101	1	15	7

différent de celui pour lequel le système NAVIGALA a été développé, car habituellement le bruit s'applique sur les images et se répercute sur les attributs de la signature et non sur les classes. Ces résultats pourraient s'expliquer par un sur-partitionnement des données comme en témoigne le nombre important de concepts dans le treillis (voir Tab. 6.11) et la présence de données aberrantes produites par ce type de bruit. NAVIGALA obtient de meilleurs résultats que CIBLe pour les bases Soybean-Small et Zoo où le nombre de classes est plus important.

En conclusion, cette étude comparative présente huit méthodes de classification basées sur l'utilisation du treillis de Galois. Les méthodes orientées sélection, dédiées à la fouille de données, nécessitent un traitement rapide, sur des données en général non bruitées. L'objectif de ces sélections est de réduire l'espace d'apprentissage selon différents critères de pertinence, tel que le nombre d'occurrence des attributs. La méthode NAVIGALA, quant à elle, est orientée navigation, et développée pour traiter efficacement des images de symboles dans un contexte bruité. Le nombre de classes traitées est potentiellement important. Nous cherchons à conserver toute l'information de l'apprentissage pour proposer de nombreux scénarii de classification et ainsi limiter l'effet du bruit sur la reconnaissance. Cette exhaustivité n'est cependant pas néfaste pour le traitement, étant donné que la complexité est largement réduite par une génération à la demande des concepts. Les données étudiées en fouille de données sont habituellement décrites par un faible nombre d'attributs. En revanche, en traitement d'images, de nombreuses signatures peuvent être extraites des images pour caractériser les objets, ainsi il en résulte un très grand nombre d'attributs. Ce type de contexte possède entre autres la particularité d'être représenté par un nombre plus petit de concepts dans le treillis par rapport au nombre de règles exactes dans une base de règles du treillis.

TABLE 6.11 – Nombre d'attributs et nombre de concepts utilisés en moyenne par NAVIGALA sur chaque base

Bases de données	Validation croisée	Nb attributs utilisés	Nb concepts du treillis
BC	10 blocs	32	6498
	5 blocs	34	6512
IR	10 blocs	13	94
	5 blocs	11	75
M2	1 bloc	16	1407
M3	1 bloc	16	1064
SS	10 blocs	6	11
	5 blocs	6	11
ZO	10 blocs	19	722
	5 blocs	18	538

TABLE 6.12 – Résultats obtenus en validation croisée sur quelques bases de données du ML Repository

Bases	Valid. croisée	Taux d'erreur de classification					
		NAVI.	RULE. [178]	CIBLe [163]	CLNB [193]	CLNN [193]	C4.5R [170]
BC	10 blocs	5.4			3.1	3.4	5.0
	5 blocs	5.5		4.6			
IR	10 blocs	7.4			5.3	5.3	4.7
	5 blocs	4.1					
M2	1 bloc	24	25.2	10.2			
M3	1 bloc	19	4.9	1.4			
SS	10 blocs	2.5					
	5 blocs	2.3		8			
ZO	10 blocs	4.0			3.9	3.9	7.8
	5 blocs	4.9		6.1			

Chapitre 7

Représentations spatiales et ontologiques

Les images de documents sont des images fortement structurées à partir desquelles il semble naturel d'extraire des primitives (traits, arcs de cercles, régions) ainsi que leur agencement spatial pour en décrire le contenu, formant ainsi une signature de l'image dite locale ou structurelle. Alors que l'analyse d'image s'intéresse en particulier à l'extraction de primitives dans un objectif de reconnaissance, leur représentation spatiale est une problématique complémentaire abordée en particulier dans les systèmes d'information géographique. En effet, la représentation de l'espace est un sujet de recherche qui suscite l'intérêt à la fois de différentes communautés scientifiques (géographie, linguistique, ...), mais aussi dans différents domaine en informatique (bases de données spatiales, systèmes d'information géographiques (SIG), raisonnement spatial, bases de données d'images...).

La section 7.1 présente quelques approches de reconnaissance d'images de symboles à l'aide d'une signature structurelle (section 7.1.1), puis décrit la *signature structurelle* que nous avons développée pour des images de symboles dans un objectif de reconnaissance (section 7.1.2)

La section 7.2 introduit l'utilisation des relations spatiales dans les SIG (section 7.2.1) et les ontologies (section 7.2.2), puis décrit l'*ontologie des lettrines* que nous avons développée dans un objectif de représentation des connaissances (section 7.2.3)

7.1 Signature structurelle et reconnaissance d'images de documents

7.1.1 Reconnaissance d'images de documents

Une signature *locale* ou structurelle vise à représenter la structure de l'image, et repose sur une décomposition de la forme en primitives élémentaires et sur leur or-

ganisation spatiale ou topologique qui rend la représentation d'un symbole possible : connexion, points de jonctions, parallèlisme, distance, orientation relative, ... Un grand nombre de signatures locales ou structurelles d'objets graphiques ont été proposées dans la littérature dans un objectif de reconnaissance d'images. On en trouve un bon état de l'art dans [208].

La mise en place d'une signature structurelle s'articule généralement autour des trois étapes que sont l'extraction de primitives à partir de l'image ; la génération des relations topologiques entre primitives ; la déduction d'une signature structurelle.

Extraction de primitives Selon le domaine considéré, les primitives peuvent être des polygones, des segments, des arcs de cercle, et plus généralement des régions de l'image. C'est pourquoi l'extraction de primitives est réalisée par un traitement d'analyse d'images adapté à la fois aux primitives à extraire, ainsi qu'aux spécificités des images. Il est essentiel d'extraire des primitives graphiques de bonne qualité, qui constitueront les briques élémentaires de la structure finale représentant le symbole.

Pour les images de symboles, les travaux existants reposent essentiellement sur une extraction de segments, comme dans [268, 218, 183], qui peuvent s'organiser en quadrilatères ou en polygones [257, 252]. Certains travaux proposent également une extraction d'arcs de cercles [269, 268]. Dans [215, 216, 240], les primitives sont des régions.

Dans son étude bibliographique, M. Delalandre [213] présente une organisation des méthodes d'extraction des primitives graphiques en sept catégories. En collaboration avec Jean-Marc Ogier et Stéphanie Guillas dans le cadre de ses travaux de thèse [40], nous avons agrémenté cette classification d'une huitième catégorie :

Détection de contours. Cette catégorie de méthodes a pour but la détection des points de contours (opérateurs morphologiques, suivi de contours ou encore décomposition en plages) puis le chaînage de ces points de contours (codage de Freeman [222]).

Squelettisation. Il s'agit d'obtenir le squelette, c'est-à -dire les axes médians d'une forme par amincissements successifs des traits, ou encore par transformée en distance.

Parcours de forme. L'objectif est également d'obtenir les axes médians de la forme par des techniques de suivi de traits, ou de détection de jonction.

Décomposition en plages. Le principe est de décomposer une forme en trois catégories de bandes verticales (ou horizontales) : plage "extrémité", plage "ligne" ou plage "jonction" afin de construire un graphe représentant l'organisation de ces plages au sein de la forme.

Segmentation en régions. Il s'agit d'identifier et d'étiqueter chaque pixel d'une forme comme appartenant à une région "extrémité", "ligne" ou "jonction" (calcul de "distances d'orientation" dans toutes les directions pour former une courbe dont les pics permettent d'identifier le type de région).

Sous-échantillonnage. Cette catégorie de méthodes consiste en l'application d'un maillage (sous-échantillonnage) sur la forme, puis en la recherche de correspondance entre chaque maille de la forme et des modèles de mailles référencées dans une bibliothèque. A partir des mailles reconnues, il est possible de déterminer la structure des primitives graphiques et leurs relations topologiques.

Composantes connexes. Il s'agit de segmenter les pixels de la forme en composantes connexes (ensemble de pixels interconnectés) pour obtenir les primitives graphiques correspondantes à ces composantes connexes et leurs relations topologiques (propagation, balayage de lignes, suivi de contours).

Transformée de Hough. [233] L'objectif de ces méthodes est de détecter dans une image un ensemble de formes géométriques simples connaissant leurs équations paramétriques, comme par exemple des droites, des cercles, des ellipses, ...

Dans le tableau synthétique 7.1,nous présentons les différentes caractéristiques de ces huit catégories de méthodes. Tout d'abord, les différentes méthodes offrent des qualités de représentation qui vont du simple ensemble de pixels à des représentations plus riches sémantiquement telles que les vecteurs qui correspondent à un trait de l'image. Elles peuvent comporter des informations sur les jonctions ou les axes médians des formes et proposent des propriétés de robustesse et d'invariance plus ou moins intéressantes.

TABLE 7.1 – Synthèse des propriétés des méthodes d'extraction de primitives graphiques

Méthodes	Qualité de la représentation	Robustesse	Invariance
Détection de contours	+ Représentation exacte - Jonctions, axes médians	++	++
Squelettisation	+ Axes médians - Distorsions de jonctions	-	+
Parcours de formes	+ Primitives vecteurs + Jonctions	+	-
Décomposition en plages	+ Représentation riche + Jonctions, axes médians	-	-
Segmentations en régions	+ Jonctions - Sémantiquement faible	++	++
Sous-échantillonnage	+ Primitives graphiques de types variés + Sémantiquement riche	-	-
Graphe de composantes	+ Primitives composantes connexes - Jonctions, axes médians	++	++
Transformée de Hough/Radon	+ Primitives de haut niveau + Jonctions, axes médians	++	+

La plupart des catégories de méthodes font l'objet d'un compromis entre qualités de représentation et propriétés de robustesse et d'invariance. Il semble ainsi que la méthode basée sur les transformées de Hough offre le meilleur compromis, de par sa représentation sémantiquement riche - extraction des traits de l'image, et ses performances en terme de robustesse et d'invariance. Grâce à son caractère global, cette méthode semble en effet pouvoir absorber différents types de bruit.

Relations spatiales Une fois les primitives extraites, leur organisation topologique rend la représentation d'un symbole possible : connexion, points de jonctions, parallélisme. Il s'agit là d'une relation binaire entre primitives qui peut s'organiser sous forme de graphe ou encore d'une matrice (matrice carrée d'adjacence), qui sont les deux représentations usuelles et équivalentes d'une relation binaire. Pour une description plus fine de l'image, les primitives et relations peuvent être attribuées (caractéristiques statistiques, coordonnées dans l'image, ... pour les primitives ; distance entre primitives, ... pour les relations topologiques).

La représentation matricielle organise l'information sous forme d'une matrice, avec les primitives en lignes et colonnes, et les relations topologiques entre primitives spécifiées dans les cases. Dans la représentation classique sous forme d'un graphe défini les noeuds représentent les primitives, et les arcs les relations. Lorsque les relations sont spécifiées pour chaque paire de primitives, alors le graphe est complet. Lorsque des attributs sont utilisés pour affiner la description, on parle de graphe attribué. Une représentation duale, avec les noeuds représentant les jonctions et les arcs les segments, est également utilisée [255, 270], notamment pour de la reconnaissance de caractères.

Dans le cas général où les primitives sont des régions, les algèbres RCC sont souvent utilisées pour représenter toutes les relations topologiques possibles entre deux régions à l'aide de 5 ou 8 relations selon que l'on considère ou non les frontières (déconnecté, superposé, inclus, contient, égale) [253, 210]. Lorsque les primitives sont des segments, on trouve dans la littérature plusieurs propositions de représentations adaptées des relations topologiques [221, 234].

Signature structurelle Une fois les primitives extraites d'un symbole, et leurs relations topologiques définies, certaines méthodes utilisent directement le graphe topologique qui les décrit pour comparer les symboles pré-segmentés par isomorphisme de graphe [252, 246, 242]. La problématique de recherche par le contenu (Content Based Information Retrieval), peut également s'envisager avec une telle approche, il s'agit alors de retrouver un symbole requête dans une image de document par recherche d'un sous-graphe dans un graphe. Cependant, la comparaison d'objets par appariement de graphes reste limitée à des données de faible dimension car il n'existe pas d'algorithme efficace garantissant une solution optimale (problème NP-complet).

C'est pourquoi une grande variété d'heuristiques au problème d'isomorphisme ont

été proposées [245]. Un grand nombre d'entre elles font l'hypothèse de l'invariance des sommets, hypothèse selon laquelle il existe une affectation de valeur aux sommets d'un graphe - très souvent son degré - qui soit invariante par isomorphisme. Il s'agit là des techniques dites de *"graph probing"* à partir desquelles des mesures de similarité entre graphes sont envisageables (distance d'édition, ...) pour mieux comparer les graphes entre eux. De telles mesures sont utilisées dans les travaux décrits dans [269, 268, 214, 215] pour comparer des symboles.

Les approches dites de *"graph embedings"* définissent un plongement du graphe dans un vecteur numérique qui constituera la signature structurelle à partir de laquelle des méthodes de reconnaissance plus classiques peuvent s'appliquer. On obtient une description du graphe qui peut être statistique (voisinage moyen, moyenne des valuations des arcs, ...) mais aussi structurelle basée sur la notion de chemins du graphe représentatifs d'une forme (sacs de chemins, histogramme de chemins). L'exploitation d'une information statistique attachée aux primitives extraites est ainsi rendue possible.

Dans le domaine de la reconnaissance, de telles approches ont été proposées, combinées avec un classifieur statistique [270, 240, 254]. Citons également les travaux dans [33] où un classifieur symbolique similaire à la méthode NAVIGALA est utilisé.

On peut également citer d'autres approches où les primitives extraites d'un symbole sont organisées en règles grammaticales [258] dans le but d'associer à une combinaison de primitives une forme constitutive du symbole : un ensemble de primitives (segments) accompagné de contraintes topologiques entre ces primitives forment la prémisse de la règle ; la conclusion correspond quant à elle à la forme décrite par ces primitives. Ainsi, trois segments connectés entre eux permettent de déduire l'existence de la forme triangle.

7.1.2 Description de la signature structurelle développée

Dans le cadre du travail de thèse de Stéphanie Guillas [40], en collaboration avec Jean-Marc Ogier, nous avons développé un signature structurelle dédiée aux symboles. La mise en oeuvre a été effectuée en deux phases : l'extraction de primitives a été développée avec l'aide de Simon Bernard [36] en 2006 ; la construction de la signature à partir de ces primitives a été implémentée avec l'aide de Mickaël Coustaty [38] en 2007.

Les symboles que nous considérons étant composés principalement de segments, nous avons développé une signature structurelle d'un symbole à partir du graphe topologique des segments. Plus précisement, cette signature correspond à un histogramme de chemins du graphe, où un chemin peut s'interpréter comme une forme constitutive du symbole.

Extraction de segments L'extraction des segments est réalisée par la transformée de Hough [233], méthode reconnue pour sa robustesse et ses propriétés d'invariance qui, de plus, possède la particularité d'extraire des segments maximaux.

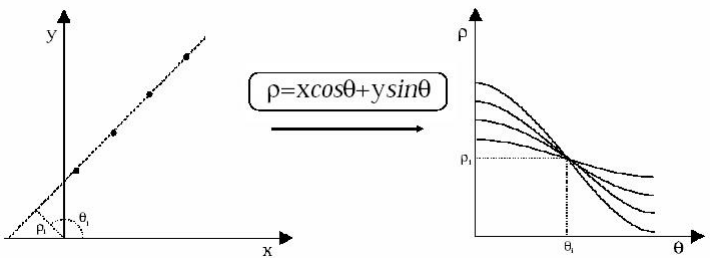


FIGURE 7.1 – Extraction de lignes par transformée de Hough

L'idée de base est que l'on peut transformer un ensemble de formes géométriques modèles (ligne, cercle, ...) d'un espace à 2 dimensions, vers un espace paramétrique dans lequel elles peuvent être représentées de façon plus compacte, et ainsi être plus facilement détectables. Plus précisément, si une forme de paramètres p est présente dans l'image, tous les pixels qui la composent seront plongés dans l'espace paramétrique vers l'unique point de coordonnées p. Ainsi, une forme de paramètres p sera détectée si elle est représentée dans l'espace paramètrique par un nombre suffisant de pixels. La Figure 7.1 illustre comment tous les points d'une droite représentés par leurs coordonnées (x, y) sont identifiés par un unique point (ρ, θ) correspondant aux coordonnées polaires de la droite définie par $\rho = x \times \cos\theta + y \times \sin\theta$. Une meilleure robustesse est obtenue en considérant le voisinage proche de (ρ, θ). Le nombre de paramètres de la forme recherchée correspond au nombre de dimensions de l'accumulateur, et influe fortement sur les temps de calculs. Ainsi, alors qu'une droite se caractérise à l'aide des deux paramètres (ρ, θ), trois paramètres sont nécessaires pour caractériser un cercle (son rayon r et les coordonnéees (x_0, y_0) de son centre).

Nous avons adapté la transformée de Hough pour détecter précisèment les bornes des segments de droite composant le symbole, contrairement aux approches classiques qui renvoient la droite portant le segment. Une détection des arcs de cercle selon la même approche, à partir des cercles détectés, est envisageable.

Relations topologiques Les relations topologiques possibles entre segments que nous utilisons, inspirées de celles proposées dans [221, 234], se limitent à cinq relations : trois relations pour des segments connectés (X, Y, V), et deux relations complémentaires pour des segments non connectés (P, et 0) (voir Figure 7.2). Pour chaque paire de segments (s, s') est ainsi précisé le type de la relation topologique entre s et s'.

Deux attibuts supplémentaires peuvent être ajoutés, et on parlera alors de triplets : il s'agit de la longueur relative entre deux segments, ainsi qu'une valeur qui peut être soit l'angle formé entre s et s' lorsqu'ils sont connectés (relations X, Y et V), sinon la distance relative entre ces deux segments. Une discrétisation de l'ensemble des valeurs possibles est cependant envisageable : discrétisation des valeurs d'angles (30°, 45°, 60°,

FIGURE 7.2 – Relations topologiques entre segments

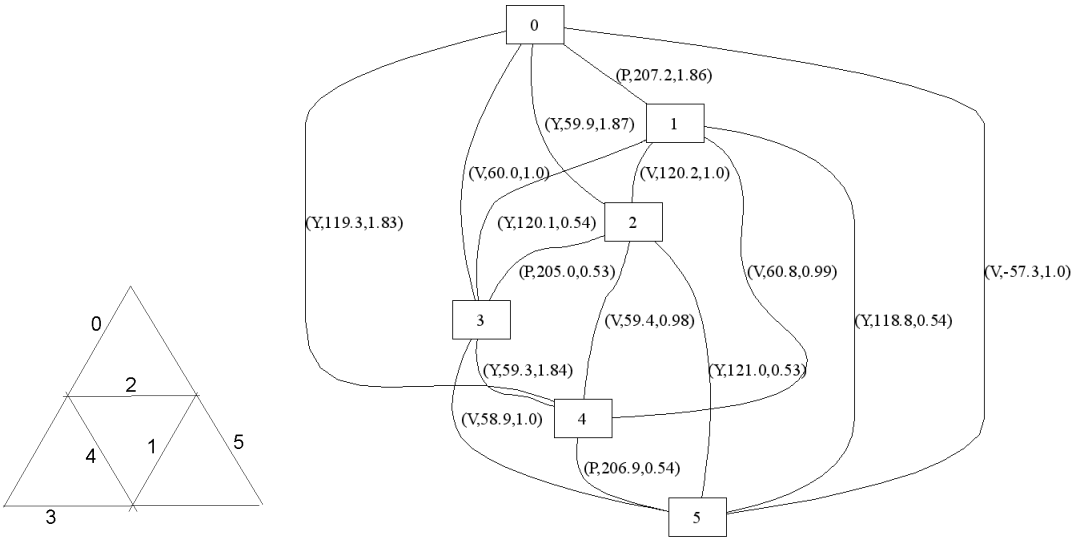
90°), discrétisation des distances entre segments parallèles (colinéaires, proches, espacés, éloignés) ou encore discrétisation des distances relatives (égaux, légèrement différents, très différents). Une analyse statistique des segments extraits pourrait permettre d'obtenir une discrétisation mieux adaptée.

Dans le graphe topologique qui s'en déduit, les noeuds correspondant aux primitives, et les arcs aux relations topologiques, chaque arc est attribué par le type/triplet de la relation topologique entre les deux segments qu'il relie. Il s'agit d'un graphe complet que l'on peut représenter par sa forme graphique, ou bien par sa matrice d'adjacence. Dans notre cas, la représentation par matrice d'adjacence est plus appropriée au calcul des chemins du graphe.

A titre d'exemple, considérons les segments de la figure 7.3(a) extraits d'un symbole. La représentation grahique du graphe topologique est représentée par la figure 7.3(b)) où chaque noeud correspondent à un segment, et chaque arc entre deux noeuds est valué par le triplet décrivant la relation topologique entre les segments associés. La matrice d'adjacence de ce graphe est représentée par la figure 7.3(b)), avec les segments en ligne et en colonne, et le triplet décrivant la relation topologique entre deux segments dans la case associée.

L'utilisation de toutes les relations topologiques entre paires de segments offre une représentation des symboles portant une certaine redondance. C'est pourquoi plusieurs restrictions des triplets considérés sont envisageables, tout en conservant une description correcte de la forme. Ainsi, il est possible de ne considérer par exemple que les relations de connexion (X,Y,V), ou encore les seules relations parallèles qui soient proches. Le graphe topologique perd alors sa propriété de complétude, et la matrice n'est pas renseignée dans sa totalité.

Signature structurelle Le calcul de la signature mis en place correspond à un histogramme de chemins, où chaque chemin du graphe est considéré comme décrivant une forme constitutive du symbole. Prenons l'exemple d'un chemin entre trois segments reliés par des arcs de type V. Un tel chemin correspond à la forme triangle, et sera décrit par "VVV". De la même façon, un "papillon" sera décrit par "PVXV". Les attributs supplémentaires des noeuds et arcs pourraient permettre de calculer diverses caractéristiques des formes : position dans l'image, aire, angles, ... La signature structurelle que nous avons développée exploite le calcul de chemins du graphe, et s'inspire des travaux proposés dans [224]. Plus précisément, elle est composée, pour chaque symbole,



(a) Segments extraits (b) Graphe topologique

	0	1	2	3	4	5
0		P 207.2 1.86	Y 59.9 1.87	V 60 1	Y 119.3 1.83	V -57.3 1
1	P 207.2 1.86		V 120.2 1	Y 120.1 0.54	V 60.8 0.99	Y 118.8 0.54
2	Y 59.9 1.87	V 120.2 1		P 205 0.53	V 59.4 0.98	Y 121 0.53
3	V 60 1	Y 120.1 0.54	P 205 0.53		Y 59.3 1.84	V 58.9 1
4	Y 119.3 1.83	V 60.8 0.99	V 59.4 0.98	Y 59.3 1.84		P 206.9 0.54
5	V -57.3 1	O 118.8 0.54	Y 121 0.53	V 58.9 1	P 206.9 0.54	

(c) Matrice d'adjacence

FIGURE 7.3 – Segments extraits par transformée de Hough sur une image de symbole et le graphe de vecteurs associé

TABLE 7.2 – Matrice d'adjacence élevée au carré

	0	1	2	3	4	5
0		4VY	2PV 2VY	2PY 1VV 1YY	2PV 2VY	2PY 1VV 1YY
1	4VY		1YY 2PY 1VV	2VY 2PV	2PY 1YY 1VV	2PV 2VY
2	2PV 2VY	1YY 2PY 1VV		4VY	2PY 1VV 1YY	2VY 2PV
3	2PY 1VV 1YY	2VY 2PV	4VY		2PV 2VY	1YY 2PY 1VV
4	2PV 2VY	1YY 2PY 1VV	2PY 1VV 1YY	2PV 2VY		4VY
5	2PY 1YY 1VV	2PV 2VY	2VY 2PV	1YY 2PY 1VV	4VY	

de la description et de la cardinalité de tous les chemins d'une longueur inférieure à un seuil, c'est-à -dire de toutes les formes constitutives du symbole composées au plus d'un nombre fixé de primitives.

En théorie des graphes, le problème quasi-similaire de l'existence d'un chemin de longueur k dans un graphe est résolu par le calcul de la kème puissance de la matrice d'adjacence du graphe, i.e. une succession de produits de matrices booléennes. Dans le cas des symboles, il s'agit non pas de déterminer l'existence d'un chemin, mais d'en calculer une description par concaténation du type des arcs qui le composent. C'est pourquoi nous avons adapté l'algorithme classique de produits de matrices booléennes à un produit de matrices de caractères qui intègre à la fois la *restriction aux chemins élémentaires* (chemins ne contenant pas plusieurs fois le même noeud), ainsi que la *factorisation de la symétrie de chaînes* (par exemple, XVPV=VPVX=PVXV).

Nous calculons les puissances de matrices jusqu'à l'ordre 4 car pour les ordres supérieurs, les agencements étudiés peuvent être trop affectés par les bruits de distorsion. Ainsi, en calculant plusieurs puissances d'une même matrice, nous obtenons un ensemble de chemins de longueurs différentes qui caractérisent le symbole. La dernière étape consiste à collecter ces chemins et compter le nombre de fois où ils sont présents pour un symbole donné. En d'autres termes, nous recherchons le nombre d'occurrences d'agencements particuliers au sein du symbole de manière générique, c'est-à-dire sans avoir pré-défini les agencements recherchés. Les matrices d'adjacence étant symétriques, on ne compte que les chemins présents dans la partie supérieure des matrices.

Si nous reprenons la matrice d'adjacence du graphe topologique de la figure 7.3(c) réduite aus seuls type des relations, à savoir X, Y, P, V et O. Le carré de cette matrice (Table 7.2) décrit tous les chemins de longueur 2 entre deux sommets, ou encore toutes

Chemins	VY	PV	PY	VV	YY
Occurrences	24	12	10	6	3

TABLE 7.3 – Exemple de signature structurelle obtenue

les formes constitutives du symbole composées de 2 segments. On peut en effet vérifier que les chemins de longueur 2 entre les segments 0 et 1 sont composés de deux arcs valués par la relation V et la relation Y. Chacun de ces chemins passe par un noeud distinct, et par conséquent sont au nombre de 4. Il y a donc 4 chemins VY de longueur 2 (4PV) entre les segments 0 et 1, description de chemins que l'on retrouve dans la case $[0, 1]$ de la matrice élevée au carrée. La table 7.3 présente la signature structurelle de ce symbole, composée du nombre d'occurences de chaque chemin de longueur 2. Au total, le symbole est constitué de 55 chemins de longueur 2, dont 24 sont des chemins VY.

La taille de la signature obtenue est déterminée par le nombre de chemins trouvés dans les symboles, qui peut varier d'un symbole à l'autre, mais aussi par le niveau de description des triplets. Une extension envisageable consisterait à limiter la taille de la signature en sélectionnant les chemins les plus discriminants.

7.1.3 Expérimentations et conclusion

Plusieurs études de la signature ont été réalisées sur la base des symboles GREC 2003 [175] réduite aux seuls symboles constitués de segments, à savoir 19 classes. Cette restriction s'explique par le fait que les arcs de cercle n'ont pas encore été intégrés dans la signature structurelle. La base d'apprentissage se compose de 10 symboles pour chaque classe : le symbole non détérioré, et un symbole par niveau de bruit. La base de reconnaissance est composée de 9 symboles par niveau de bruit et par classe, d'où 1539 symboles pour 19 classes.

Analyse statistique

Une analyse statistique des triplets topologiques fait ressortir certaines particularités des segments extraits. Nous avons pu observer la forte présence des relations V, P, et O. Les signatures structurelles comportant les relations X et Y seront par conséquent les plus discriminantes. Nous avons également noté la présence conséquente des angles droits dans les symboles, ce qui ne semble pas aberrant puisque l'on rencontre un nombre important de carrés et de rectangles. Enfin, une grande majorité des segments extraits sont de taille égale.

La taille de la signature augmente largement en fonction de la longueur des chemins (table 7.4). En prenant le triplet complet, pour les chemins de longueur 1, la taille de la signature passe à 63. C'est pourquoi une limitation envisageable serait de sélectionner les chemins les plus discriminants. Une étude des fréquences d'apparition des chemins

TABLE 7.4 – Taille des signatures obtenues pour différentes longueurs de chemins

Longueur des chemins (triplets restreints aux types de relation)	1	2	3	4
Taille des signatures	7	28	161	759

Longueur des chemins	1	2	1 et 2	3	2 et 3	1, 2 et 3
Taux de reconnaissance (1^{er} rang)	70%	70.2%	73.8%	69.9%	67%	68.9%
Taux de reconnaissance (n^{me} rang)	86.5%	86.9%	87.9%	86.5%	83.9%	83.6%
Nb de caractéristiques	53	206	259	762	968	1021

TABLE 7.5 – Résultats expérimentaux avec un triplet partiel

dans les symboles fait ainsi apparaître, comme l'on s'y attendait, que les combinaisons LP, LV et VV qui correspondent à une succession d'angles à 45° ou 90° sont les plus représentées.

Expérimentations

Une validation de notre signature a été menée dans un contexte de classification. Nous avons pour cela utilisé la méthode NAVIGALA, ainsi que le classifieur LINEAR-SVM. Plusieurs expérimentations ont été menées.

Nous avons tout d'abord comparé les taux de reconnaissance obtenus en utilisant un triplet partiel avec des chemins de longueur au plus 3, et un triplet complet avec des chemins de longueur au plus 2. Les tables 7.5 et 7.6 présentent les résultats obtenus. On peut tout d'abord observer que les résultats sont relativement proches. Etant donné que les triplets partiels ne sont composés que du type de la relation, alors que les triplets complets considèrent plus d'attributs la décrivant, on peut en déduire que le type de relation est une information pertinente. Cependant, la caractériser par des attributs supplémentaires permet d'augmenter les résultats de 5% en moyenne.

Longueur des chemins	1	2	1 et 2
Taux de reconnaissance (1^{er} rang)	79%	69.5%	71.5%
Taux de reconnaissance (n^{me} rang)	90.4%	85.3%	87.7%
Nb de caractéristiques	132	790	922

TABLE 7.6 – Résultats expérimentaux avec un triplet partiel

Deux types de taux de reconnaissance sont indiqués. Le taux de reconaissance de 1^{er} rang correspond au pourcentage de symboles correctement classé dans un concept

terminal pur - i.e. un concept terminal dont les objets appartiennent à la même classe. Le taux de reconnaissance de n^{me} rang correspond au pourcentage de symboles correctement classés dans un concept terminal contenant au plus n classes parmi lesquelles la classe du symbole.

La table 7.7 présente le nombre de classes par concept terminal. Nous pouvons ici observer que n est inférieur ou égale à 3 dans tous les cas, et que 93% des concepts terminaux sont purs. On peut également remarquer que les taux de reconnaissance obtenus avec le n^{me} rang sont supérieurs de 15,92% à ceux obtenus avec les taux de reconnaissance de premier rang.

Nb de classes par concept terminal	Nb de concepts terminaux	% du nombre total de concepts terminaux
1	81	93.1%
2	5	5.74%
3	1	1.14%

TABLE 7.7 – Distribution des classes dans les 87 concepts terminaux du treillis

Ce phénomène s'explique par la fait que notre signature structurelle est moins discriminante qu'une signature statistique. Cependant, une étude comparative réalisée avec Navigala entre la signature structurelle et une signature statistique fait apparaître que ces deux signatures sont complémentaires. En effet, 94,5% des symboles mal-reconnus avec la signature statistique sont correctement reconnus avec la signature structurelle. Ainsi, la signature structurelle pourrait donc être avantageusement combinée avec une signature statistique dans un système utilisant NAVIGALA comme classificateur.

Les résultats présentés dans la table 7.5 montrent également que l'utilisation de chemins de longueur 3 fant baisser les taux de reconnaissance. Ceci peut s'expliquer par le fait que les chemins de longueur 3 portent une information redondante avec celles des chemins de longueurs 1 et 2. Cette redondance augmente la corrélation entre les attributs de la signature, et a un impact négatif sur le taux de reconnaissance.

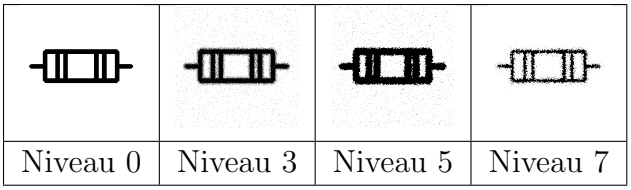


FIGURE 7.4 – Différents niveaux de dégradation pour un même symbole

La troisième série de test a été effectuée afin de mettre en évidence la robustesse de la signature, et la stabilité des résultats envers les différents niveaux de dégradation. La table 7.8 présente les résultats obtenus pour chaque niveau de dégradation. On peut observer que les taux de reconnaissance sont similaires, à l'exception des niveaux 5 et

7 pour lesquels le niveau de bruit est très élevé. En effet, les niveaux 5 et 7 sont très fortement perturbés par un épaississement et un amincissement des traits. L'épaississement provoque une sous-segmentation, et les segments proches sont fusionnées, alors que l'amincissement provoque une sur-segmentation et donc une perte de connexité. Même si la transformée de Hough peut faire face à certains phénomènes, ces défauts sont amplifiés par le prétraitement de squelétisation. Nous pouvons également remarquer que le taux de reconnaissance pour tous les niveaux de dégradation, à l'exception des niveaux 5 et 7, est 92,63 %.

Niveau de dégradation	1	2	3
Taux de reconnaissance (n^{me} rang)	96.5%	94.7%	93%
Niveau de dégradation	4	5	6
Taux de reconnaissance (n^{me} rang)	90%	80.7%	91.2%
Niveau de dégradation	7	8	9
Taux de reconnaissance (n^{me} rang)	84.2%	90%	93%

TABLE 7.8 – Résultats expérimentaux par niveau de dégradation, en utilisant un triplet partiel de longueur 1

Enfin, nous avons réalisé une validation croisée en utilisant le triplet partiel des chemins de longueur 1, et un découpage en 10 blocs. Les résultats sont présentés dans la table 7.9. Avec un classement de premier rang, les résultats obtenus sont similaires à ceux obtenus avec un ensemble d'apprentissage plus petit (voir table 7.5). Nous pouvons en déduire que la combinaison de la signature structurelle et de NAVIGALA permet d'extraire la structure constitutive des symboles, quel que soit le niveau de dégradation qui est appliqué. Les taux de reconnaissance augmentent de 9% avec la classification de n^{me} rang. Ce phénomène peut s'expliquer par les différences dans le protocole expérimental. En effet, le nombre d'exemples de la base d'apprentissage est plus élevé en validation croisée, et le treillis apprend alors les différents types de bruit.

	Taux moyen de reconnaissance	Déviation standard
Taux de reconnaissance (1^{er} rang)	69.78%	3.43%
Taux de reconnaissance (n^{me} rang)	94.97%	3.31%

TABLE 7.9 – Validation croisée avec un triplet partiel est des chemins de longueur 1

Une dernière expérimentation est consacrée à la comparaison entre l'utilisation de NAVIGALA et d'un classifieur linéaire SVM. La table 7.10 présente les résultats obtenus. Afin d'avoir un unique résultat pour NAVIGALA, nous considérons la classe majoritaire du concept terminal comme résultat de la classification. On observe que Navigala permet d'obtenir de meilleurs résultats que ceux obtenus avec le classifieur liéaire SVM.

Longueur de chemins	1	2	1 et 2	3
Linear-SVM	71.54%	69.33%	72.06%	69.33%
Navigala (vote majoritaire)	77.06%	73.88%	77.45%	69.9%

TABLE 7.10 – Comparaison entre l'utilisation de NAVIGALA et du classifieur Linear-SVM avec un triplet partiel.

Enfin, pour préciser ce comparatif avec le classifieur linéaire SVM, nous avons réalisé un dernier test en utilisant un petit ensemble de symboles en apprentissage (1 % de la base des symboles). Il s'agit là d'une situation que l'on retrouve dans de nombreux contextes applicatifs où un seul modèle par classe est fourni. Ce test a été effectué en utilisant des chemins de longueur 2 et un triplet partiel. Le taux de reconnaissance obtenu par le classifieur linéaire SVM est de 55,56%, tandis que celui de NAVIGALA est de de 57,25 %. La différence entre ces deux taux de reconnaissance permet de renforcer l'idée que la méthode NAVIGALA, et de façon plus générale le treillis des concepts, est capable de conceptualiser l'espace de reconnaissance avec moins d'éléments en apprentissage que les classificateurs statistiques.

Conclusion et perspectives

La signature structurelle mise en place est une méthode générique qui offre une représentation haut niveau des symboles. En effet, les différentes formes définies par des agencements spatiaux entre les segments utilisés pour caractériser les symboles ne sont pas prédéfinis (généricité) et leur taille est paramétrable (représentation haut niveau). Plus leur taille est importante, plus la description des symboles devient complexe et discriminante, mais dans le même temps, plus la robustesse aux distorsions est faible [257]. Il est donc nécessaire de trouver le bon compromis entre niveau de description et robustesse. C'est pourquoi une analyse statistique des symboles considérés reste indispendable pour un bon paramétrage.

Une signature structurelle dédiée aux symboles en fournit une représentation d'un niveau de description plus élevé qu'une signature statistique, et peut par conséquent s'utiliser dans un cadre plus large. En effet, la structure de graphe est particulièrement bien adaptée pour de la recherche en contexte, par isomorphisme de sous-graphes [246, 238] par exemple. Des méthodes de hiérarchies de graphes ont également été introduites en reconnaissance de caractères où un graphe est associé à chaque caractère [241].

Par ailleurs, alors que l'utilisation d'une signature statistique a le plus souvent un objectif de classification ou d'indexation, une signature structurelle décrivant de façon générique les formes constitutives d'un symbole permet d'envisager une recherche d'information plus sophistiquée : recherche de tous les symboles contenant une forme requête, ou encore recherche des formes communes à un ensemble de symboles requêtes. De plus, les images de symboles sont issues d'un domaine souvent porteur d'une connais-

sance experte forte. C'est pourquoi l'utilisation d'un langage de description du domaine pourrait permettre une recherche par requête intégrant à la fois une information de bas niveau (la description de formes), mais également la sémantique du domaine. Les technologies de types ontologies, et plus particulièrement les logiques de description sous-jacente sur lesquelles elles reposent, nous semblent tout indiquées pour une telle approche.

7.2 Ontologie des relations spatiales

7.2.1 Relations spatiales et SIG

Les SIG offrent une large gamme de modèles et outils adaptés à la représentation spatiale [263], ainsi que la possibilité d'associer des informations attributaires aux primitives telles que leur nature (route, voie ferrée, forêt, etc.) ou toute autre information contextuelle (nombre d'habitants, type ou superficie d'une commune par ex.). La distinction y est faite entre le *modèle raster*, encore appelé modèle matriciel ou modèle continu, où l'espace est décomposé en cellules définies selon un maillage régulier et plus ou moins précis ; et le *modèle vectoriel*, encore appelé modèle trait ou modèle discret, où l'espace est composé de lignes définies comme une succession de points en coordonnées réelles (x, y), et qui peuvent définir des polygones. Une image vectorielle (ou image en mode trait) est une image numérique composée d'objets géométriques individuels (segments de droite, polygones, arcs de cercle, etc.) définis chacun par divers attributs de forme, de position, de couleur, etc. Elle se différencie de cette manière des images matricielles (ou « bitmap »), dans lesquelles on travaille sur des pixels.

Objets spatiaux Les objets spaciaux les plus génériques sont des régions, classiquement définies comme des polygones - série de lignes - ou encore des ensembles de points. Pour une meilleure interopérabilité, le consortium OGC (Open Geopspatial Consortium) a pour but d'élaborer des standards de modélisation dans le domaine géographique, de façon équivalente à celle existant pour le texte par exemple. Au niveau vectoriel, le *standard SFS* (Simple Feature Interface Standard) proposé par l'OGC en 1999 définit une classe abstraite principale (classe `Geometry`) pour représenter des objets spatiaux quelconques ainsi que l'illustre la figure 7.5. Cette classe peut se dériver en un point (classe `Point`), une courbe (classe `Curve`), ou encore une surface (classe `Surface`), elles-mêmes définies à partir de lignes (classe Line).

Relations spatiales On distingue plusieurs types de relations spatiales : les relations topologiques, les relations de direction et les relations de distance. Les *algèbres RCC-8* [210] (Region Connection Calculus), définies à partir de la relation de connexion introduite dans [209], sont souvent utilisées pour modéliser les relations topologiques

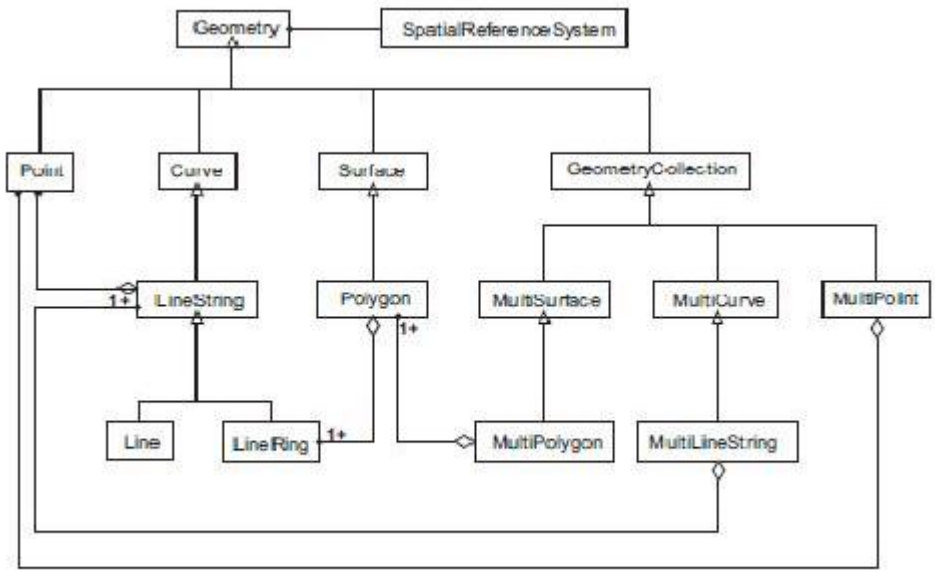


FIGURE 7.5 – Modèle de classe des types spatiaux dans le standard SFS de l'OGC

entre régions. Elles sont composées de huit relations exhaustives (toute relation entre deux régions est décrite) et mutuellement exclusives (cf Figure 7.6).

EQ(x,y) : x est égal à y.

NTPP(x, y) : x est une partie propre non tangentielle de y.

TPP(x, y) : x est une partie propre tangentielle de y.

NTPP-1(x, y) : x possède pour partie propre non tangentielle y.

TPP-1(x, y) : x possède pour partie propre tangentielle y.

PO(x, y) : x recouvre partiellement y.

EC(x, y) : x est connexe à y.

DC(x, y) : x est déconnecté de y.

Inférence spatiale Il existe plusieurs approches pour caractériser et calculer les relations RCC-8, que ce soit pour des données vectorielles où raster. Dans [243], chaque région x est définie qualitativement par deux ensembles de points : l'intérieur x^o et la frontière $\delta(x)$). Les quatre intersections $\delta(x) \cap \delta(y)$, $x^o \cap y^o$, $\delta(x) \cap y^o$ et $x^o \cap \delta(y)$ suffisent à déterminer les huit relations topologiques de l'algèbre RCC-8. Une description plus fine, proposée dans [220] introduit également le complément x^{-1} d'une région. Citons également les travaux de [203, 244] où les relations RCC-8 sont définies à partir de trois opérations $x^o - y^o$, $x^o \cap y^o$ et $\delta(x) \cap \delta(y)$ qui font intervenir l'intersection, mais aussi la différence ensembliste.

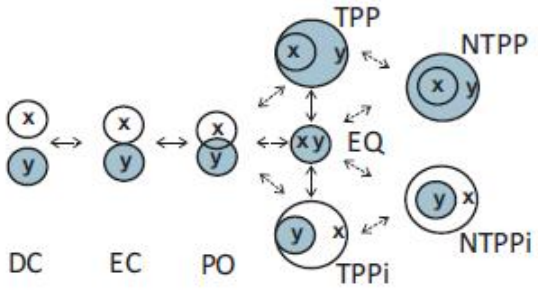


FIGURE 7.6 – Relations spatiales de l'algèbre RCC-8

Dans un SIG, il n'est pas envisageable de calculer la relation topologique entre chaque paire de primitives, leur nombre étant souvent trop important. C'est pourquoi les caractéristiques algébriques des relations utilisées (symétrie, composition, transitivité,...) sont fondamentales dans la mise en oeuvre d'un mécanisme d'inférence topologique [225]. Les règles de composition sont particulièrement importantes, car, à l'aide d'un tableau de composition, elles permettent de déduire la relation topologique entre deux régions x et z à partir de celles entre x et y, et entre y et z. Cependant, les relations inférées sont souvent imprécises, pouvant correspondre à plusieurs relations possibles, et la réitération de ce mécanisme de composition rend l'inférence topologique complexe. Il s'agit d'un problème NP-complet pour lequel, sous certaines conditions, il existe des algorithmes d'inférence polynomiaux. Plusieurs outils proposant l'inférence spatiale sont disponibles (Pellet Spatial, Geo-swrl, Oracle Spatial,...). Le lien entre les opérations de calcul et les relations RCC-8 a été formalisé dans [204, 205, 195] pour une aide au raisonnement spatial avec un treillis de concepts.

7.2.2 Représentation ontologique des connaissances

Les images de document sont des images issues d'un domaine souvent porteur d'une connaissance métier sémantiquement forte (lettrines, symboles, plan cadastral) qu'il semblerait naturel d'intégrer aux données dans un objectif de représentation de l'information. Les technologies de types *ontologies*, issues du champ de la représentation des connaissances et de raisonnement et introduites récemment dans le domaine du web sémantique pour permettre d'organiser les données et connaissances d'un domaine dans le but de les partager, diffuser et actualiser, fournissent un cadre méthodologique à la fois pertinent et novateur pour traiter des images de document.

Une ontologie permet une description à la fois structurelle et un ensemble de données hétérogènes, mais aussi des connaissances sous-jacentes. Les fondements des ontologies reposent sur les logiques de description, langages de représentation des connaissances utilisés pour décrire un domaine applicatif de façon structurée et lisible, permettant ainsi une recherche d'information par requête, en interaction avec l'utilisateur.

Plusieurs méthodologies de construction d'une ontologie sont proposées dans la littérature, et nécessitent avant tout une bonne compréhension du domaine. Alors que les premières ontologies sont construites à la main, de récents travaux s'intéressent à des méthodes semi-automatiques de construction d'une ontologie directement à partir des données hétérogènes, notamment dans le domaine de la fouille de textes. L'analyse formelle des concepts est une méthodologie d'analyse et de fouille de données particulièrement bien adaptée.

Logique de description

Les fondements des ontologies reposent sur les *logiques de descriptions*, sous-langages de la logique du premier ordre qui proposent un formalisme de représentation des connaissances pour représenter et interroger la connaissance d'un domaine applicatif de façon structurée et lisible.

Les premiers travaux autour des logiques de description ont été initiés dans le but de fournir des fondements logiques pour une meilleure représentation sémantique. En effet, bien que les premiers formalismes de représentation des connaissances - les réseaux sémantiques, les frames - soient reconnus pour leur lisibilité, l'absence de mécanismes de raisonnements théoriques et de fondements logiques est un frein à leur exploitation.

Le langage KL-ONE [207] est considéré comme le langage fondateur de représentation des connaissances qui, initialement nommés *langages de représentation "KL-ONE-like"*, se définissent comme des fragments de la logique du premier ordre caractérisés par un pouvoir d'expressivité sémantiquement fort, mais surtout par des résultats de décidabilité des raisonnements. L'adoption du langage OWL [230] - langage basé sur les logiques de description - comme langage standard de description d'ontologies pour le web sémantique, a plus récemment mis en avant les logiques de description.

Définition Les logiques de descriptions introduisent les notions fondamentales de *concepts* et de *rôles*. Un concept est un prédicat unaire - par exemple `Man(X)` - alors qu'un rôle est un prédicat binaire - par exemple `hasChild(X,Y)`. Elles reposent sur un formalisme permettant des descriptions de *concepts* et de *rôles* à partir :

- de concepts atomiques,
- de rôles atomiques,
- d'individus,
- de constructeurs.

Les différentes logiques de description se distinguent selon les constructeurs utilisés. La table 7.11 décrit les logiques $\mathcal{EL}$, $\mathcal{ALE}$, $\mathcal{ALC}$ et $\mathcal{ALCN}$ et $\mathcal{ALQ}$ par les constructeurs qui les définissent. Il s'agit là de logiques de description pour lesquelles les rôles sont atomiques.

Toute combinaison d'opérateurs permet de définir une logique de description, il en existe un nombre potentiellement élevé. Pour les distinguer, chaque extension par ajout

Constructeur		**Logiques de description**				
Description	Syntaxe	$\mathcal{EL}$	$\mathcal{ALE}$	$\mathcal{ALC}$	$\mathcal{ALCN}$	$\mathcal{ALQ}$
négation d'un concept	$\neg C$			×	×	
conjonction de concepts	$C \sqcap D$	×	×	×	×	×
disjonction de concepts	$C \sqcup D$			×	×	
restriction existencielle (*en relation r exclusive avec des instances d'un concept*)	$\exists r.C$	×	×	×	×	×
restriction de valeur (*en relation r avec au moins une instance d'un concept*)	$\forall r.C$		×	×	×	×
concept le plus spécifique	$\top$	×	×	×	×	×
concept le plus général	$\bot$		×	×	×	×
restriction minimale (*au moins n fois en relation r*)	$\leq nr.C$				×	×
restriction maximale (*au plus n fois en relation r*)	$\geq nr.C$				×	×

TABLE 7.11 – Constructeurs pour les logiques $\mathcal{EL}$, $\mathcal{ALE}$, $\mathcal{ALC}$, $\mathcal{ALCN}$ et $\mathcal{ALQ}$

de nouveaux constructeurs est indiquée par une lettre :

$\mathcal{R}+$ **:** possibilité de décrire des transitivités de rôles.

$\mathcal{H}$ **:** possibilité de décrire une hiérarchie de rôles.

$\mathcal{O}$ **:** possibilité de décrire des classes par les individus qui la composent (types énumérés).

$\mathcal{I}$ **:** possibilité de décrire des rôles complémentaires. Par exemple, $\texttt{hasChild}^{-1}$.

$\mathcal{N}$ **:** possibilité d'intégrer des restrictions nominales. Par exemple, $\leq 2\texttt{hasChild}$.

$\mathcal{Q}$ **:** possibilité d'intégrer des cardinalités existentielles.

$\mathcal{F}$ **:** possibilité d'intégrer des cardinalité typées (rôles fonctionnels). Par exemple, $\leq 2\texttt{hasChild Male}$.

Ainsi, $\mathcal{ALCHIQ}$, également nommée $\mathcal{SHIQ}$, est une extension de la logique $\mathcal{ALC}$ avec des rôles non atomiques, transitifs, hiérarchiques et complémentaires avec restriction nominales existentielles.

Les logiques de description ont pour point commun de décrire une *base de connaissance* d'un domaine. La séparation est faite entre le *niveau structurel* appelé T-Box et le *niveau factuel* appelé A-Box :

T-Box : La T-Box est constituée de *descriptions de concepts* ou de classes, et de *description de rôles* ou de relations qui sont des formules logiques composées de concepts atomiques, de rôles atomiques, de constantes et des constructeurs de la logique utilisée.

A-Box : La A-Box est quant à elle constituée d'assertions permettant de peupler les concepts et rôles de la T-Box par des individus ou données. Une *assertion de concept* est une expression de la forme $C(a)$ où C est un concept, et a un individu. Une *assertion de rôle* est une expression de la forme $r(a, b)$ où r est un rôle et a et b sont deux individus. L'ensemble des individus appartenant à une classe correspond à l'*extension de la classe*.

Mécanisme d'inférence Les logiques de description ne sont pas seulement utilisées pour exprimer des définitions et des assertions de concepts et de rôles. Elles proposent également des *mécanismes de raisonnement* permettant de déduire de la connaissance implicite à partir de la connaissance explicitement formalisée. Ces raisonnements se construisent à partir de quatre inférences logiques :

Subsomption : Il s'agit de tester si un concept D est plus général qu'un concept C. On dit alors que D *subsume* C, ce qui est noté $D \sqsubseteq C$. En d'autres termes la subsomption permet de vérifier que les propriétés du concept C sont également des propriétés du concept D.

Satisfiabilité : Il s'agit de vérifier s'il n'existe pas de concept dont l'extension est vide, i.e. il existe au moins un individu membre pour chaque concept. Satisfiabilité et subsumption sont des problématiques liées.

Consistance : Il s'agit de vérifier la cohérence entre la A-Box et la T-Box, plus précisément entre les assertions de la A-Box et les descriptions de la T-Box. En d'autres termes, la consistance permet de vérifier si un individu membre d'une classe par assertion en vérifie bien les propriétés.

Instance : Il s'agit de vérifier si, selon ses propriétés, un individu - ou une instance - est membre des classes définies par ces même propriétés.

Ces inférences sont la base de tout raisonnement sur la T-Box ou sur la A-Box. Ainsi, la recherche de tous les individus membres de l'extension d'un concept donné, ou encore de tous les concepts dont l'extension contient un individu donné, sont des mécanismes de raisonnements très utilisés, construits à partir de ces inférences. Citons également le raisonnement *par classification* dont le but est d'identifier la hiérarchie de concepts par subsomption, et en particulier le concept le plus spécifique pour un individu, en fonction de ses propriétés.

La *satisfiabilité* est reconnue comme le problème d'inférence principal en logique de description, les autres problèmes d'inférence s'y rapportant par une transformation polynomiale. L'étude du problème de satisfiabilité a fait l'objet de nombreux travaux, sa complexité pouvant être aussi bien polynomiale que NP-complète selon la logique de description utilisée et les caractéristiques de la T-Box. Bien entendu, ce problème est polynomial pour des logiques de description simples et très peu expressives, et devient NP-complet pour les logiques de description les plus expressives. D'où l'apparition de

variantes pour lesquelles le problème de subsomption devient polynomial. Une synthèse des complexités de raisonnement est proposée dans [217].

Les algorithmes de raisonnement s'appuient sur une variante des tableaux sémantiques tels qu'ils sont introduits pour la logique du premier ordre [247]. C'est pourquoi l'ensemble des algorithmes est appelé *algorithmes de tableaux* ("tableaux-based algorithms") dans la communauté. On trouvera dans [200, 231] une synthèse de ces algorithmes de tableaux.

Le choix d'une logique de description est donc un compromis entre expressivité de son formalisme, et complexité du raisonnement. Cependant, il existe des outils de raisonnements optimisés et efficaces - FaCT [229] (et son successeur FaCT++ [265]), Racer [227] (et son successeur RacerPro) Pellet [264] - qui, malgré une complexité exponentielle dans le pire des cas, se comportent bien en pratique pour des logiques de description très expressives. Une liste exhaustive des outils existants est maintenue sur [259]. L'incomplétude d'un algorithme de subsomption devient plus une caractéristique de la logique utilisée, qu'un défaut algorithmique.

Ontologies Plusieurs définitions d'une *ontologie en informatique* sont proposées dans la littérature [266]. Pour Gruber [226] *"une ontologie est la spécification formelle et explicite d'une conceptualisation partagée d'un domaine de connaissance"*. Le langage OWL, introduit en 2003 [230] dans le domaine du web sémantique [206], a été adopté en 2004 par la consortiume W3C [249] comme standard des langages d'ontologies pour permettre d'organiser les connaissances d'un domaine dans le but de les partager, diffuser et actualiser, dans une meilleure interopérabilité. La langage OWL permet une plus grande capacité d'interprétation et de raisonnements grâce à une sémantique formelle qui repose sur les logiques de description. Il permet ainsi :

1. de formaliser un domaine en définissant des classes - ou concepts - et propriétés - ou rôles - de ces classes,
2. de définir des individus et des propriétés les concernant,
3. de raisonner sur ces classes et individus.

Soulignons que le langage OWL utilise les terminologies de classes et de propriétés, alors qu'en logiques de descrition, il s'agit de concepts et de rôles. Le langage OWL se décline en trois sous langages - OWL Lite, OWL DL et OWF Full - offrant des capacités d'expression croissantes, et en lien hiérarchique - toute ontologie OWL Lite valide est une ontologie OWL DL valide, qui est elle-même une ontologie OWL Full valide.

OWL Lite : OWL lite est le sous langage de OWL le plus "simple". Il permet de spécifier une hiérarchie de concepts, et des mécanismes de contraintes simples. OWL Lite est adapté, par exemple, aux migrations rapides entre bases de connaissances.

OWL DL : OWL DL est plus complexe que OWL Lite. Fondé, comme son nom l'indique, sur les logiques de descriptions, il garantit la décidabilité et la complétude

des systèmes de raisonnements de l'ontologie. En effet, toutes les inférences sont décidables, donc calculées en une durée finie.

OWL Full : OWL Full est la version la plus complexe d'OWL, mais également celle qui permet le plus haut niveau d'expressivité, mais sans garantie de calculs. OWL Full est destiné aux situations où il est important d'avoir un haut niveau de capacité de description.

Plusieurs outils d'édition d'une ontologie sont disponibles (Protégé [237], Jena [235]), qui manipulent une ontologie selon divers formats (RDF, RDFS, N3, ...).

Logiques de description et AFC Plusieurs méthodologies de construction d'une ontologie sont proposées dans la littérature, et nécessitent avant tout une bonne compréhension du domaine. Alors que les premières ontologies sont construites à la main, de récents travaux s'intéressent à des méthodes semi-automatiques de construction d'une ontologie directement à partir des données hétérogènes. L'analyse formelle des concepts est une méthodologie d'analyse et de fouille de données particulièrement bien adaptée.

En effet, on retrouve la même notion de *concept* à la fois en AFC et dans les logiques de description, avec cependant des différences majeures en terme de formalisme. En AFC, un concept est un ensemble maximal d'objets possédant des attributs en commun, défini formellement à partir d'un contexte où les objets y sont complétement décrits par l'ensemble des attributs qu'ils possèdent. Les logiques de description définissent un concept par une expression logique construite à l'aide des constructeurs de la logique utilisée, puis les individus sont associés aux concepts par des assertions, ou par inférence. On peut identifier ici deux différences majeures entre ces deux approches qui manipulent la même notion avec des formalismes différents :

Hypothèse du monde : L'AFC fait l'hypothèse du *monde clos*, les concepts se définissent à partir d'un ensemble d'objets aux propriétés connues a priori. A l'inverse, les logiques de descriptions font l'hypothèse de *monde ouvert*, permettant ainsi une description partielle des objets, leur appartenance à un concept pouvant se déduire par inférence, permettant ainsi d'identifier de nouvelles propriétés.

Description des objets : En AFC, les objets sont décrits par un ensemble d'attributs, i.e. par un ensemble de propriétés atomiques, alors que les logiques de description permettent des descriptions des objets à l'aide d'une formule logique.

Ces dernières années, de nombreux travaux ont cherché à unifier ces deux formalismes. Une synthèse récente en est proposée dans [261, 262]. Deux types d'approches s'y distinguent selon qu'elles visent à enrichir l'AFC à l'aide du formalisme proposé par les logiques de description, ou à l'inverse, à enrichir les logiques de description à l'aide des outils proposés par l'AFC.

Enrichissement de l'AFC par des méthodes des logiques de description :

Exploration d'attributs enrichie [251]. L'exploration d'attributs, méthode issue de l'AFC [223], vise à acquérir des connaissances d'un domaine de manière interactive en questionnant un expert. Il s'agit d'un processus itératif où, à chaque étape, l'expert est questionné sur des implications possibles entre attributs. En cas de réponse négative, il doit alors fournir un contre-exemple qui est intégré au contexte. Une nouvelle question lui est ensuite posée. Les questions sont les implications de la base canonique du treillis, ce qui garantit un ensemble de questions à la fois minimal et complet. Dans [251], un langage des logiques de description est utilisé pour enrichir les questions. Le raisonneur est utilisé en tant qu'expert pour y répondre.

Analyse relationnelle des concepts [256]. L'analyse relationnelle des concepts est une extention de l'AFC introduite en 2007 [256] pour analyser des objets décrits par des attributs relationnels. Il s'agit d'intégrer une description relationelle des objets, i.e. des prédicats binaires qui peuvent s'interpréter comme des rôles en logique de description. Chaque relation entre deux ensembles d'objets est manipulée comme un contexte. Le principe de l'ARC consiste à construire le treillis de concepts de chaque contexte initial, puis à enrichir chaque contexte, et à réitérer ce processus. Le mécanisme d'enrichissement est simple. Considérons deux contextes C_1 et C_2 décrivant deux relations, la première entre deux ensembles d'objets A et B, la seconde entre les objets de B et un ensemble d'objets C. Alors le contexte C_1 est enrichi avec les concepts du treillis de C_2, chaque objet de B sera en relation avec les concepts le contenant. Le treillis de concepts du contexte C_1 enrichi fera ainsi apparaître à la fois des objets de A, de B et de C. Le processus est réitéré jusqu'à obtention d'un unique treillis de concepts qui intègre tous les ensembles objets initiaux en respectant les relations qu'ils peuvent avoir. Un raisonneur peut éventuellement être utilisé pour vérifier la consistance des concepts ainsi obtenus.

Analyse logique des concepts [55]. L'Analyse Logique des Concepts est une extension de l'AFC introduite en 2004 [55] pour permettre une description d'objets par une formule logique. La connexion de Galois entre l'ensemble des objets et l'espace des descriptions est alors garantie à condition que la logique de description utilisée soit munie d'un élément maximal $\top$ et d'une relation de subsomption. Le treillis de Galois est alors exploité par un système d'information logique (SIL) pour proposer une recherche d'information interactive avec l'utilisateur, par requête, mais aussi par navigation. A chaque requête de l'utilisateur correspond un concept du treillis dont les objets sont le résultat de la requête. Ce concept, mais aussi ses prédecesseurs et successeurs immédiats, sont alors calculés selon le mécanisme de génération à la demande. L'utilisateur peut alors affiner sa recherche en reformulant sa

requête, mais aussi en navigant dans les concepts proposés.

Enrichissement des logiques de description par des méthodes de l'AFC :

Hiérarchie de subsomption des concepts[197]. L'exploration d'attributs est ici utilisée pour construire un contexte dont les attributs sont les concepts définis dans la T-Box, et les objets sont les contre-exemples donnés par l'expert au cours du processus d'exploration d'attributs. Les questions posées sont donc des implications entre concepts. Le treillis de concepts du contexte ainsi obtenu est isomorphe à la hiérarchie des concepts de la T-Box. Des améliorations ont été proposées [201] de manière à pouvoir utiliser les raisonneurs de la logique de description utilisée.

Hiérarchie de subsomption des plus grands concepts communs [198]. La hiérarchie des concepts est ici construite de façon ascendante, automatiquement à partir de données. A chaque étape, il s'agit d'abord de calculer les *concepts les plus spécifiques* décrivant un ensemble d'objets, puis de calculer un sous-concept commun à ces concepts spécifiques pour en affiner la description. L'exploration d'attributs est utilisée pour calculer la hiérarchie des sous-concepts communs des concepts spécifiques, permettant ainsi d'identifier un plus grand sous-concept commun pertinent.

7.2.3 Description de l'ontologie des lettrines développée

La mise en oeuvre d'une ontologie des lettrines a été réalisée en collaboration avec Alain Bouju et Georges Louis, avec l'aide de Thangh Le Ngoc dans le cadre de son stage recherche [42], et de Mickaël Coustaty dans le cadre de ses travaux de thèse en cours.

Cette ontologie intègre à la fois la connaissance métier des historiens ainsi que des informations sur des régions de l'image, et peut ainsi être interrogée sur ces deux types de propriétés. Nous l'avons enrichie de règles logiques pour une annotation sémantique de certaines régions de l'image. Il s'agit là d'un mécanisme d'ancrage sémantique des régions par inférence de règles, permettant de réduire le *fossé sémantique* entre description pixellaire et description sémantique d'une image.

Images de lettrines

Les images de lettrines sont des images graphiques que l'on retrouve dans des documents anciens des $XV^{\text{ème}}$ et $XVI^{\text{ème}}$ siècles. La Figure 7.7 en donne un exemple. Il s'agit d'une lettre décorée, en début des paragraphes des livres de cette époque, obtenue à l'aide de tampon en bois, sculpté à la main. Composée principalement d'une lettre en son centre, une lettrine se caractérise également par un fond qui peut être décoratif, ou bien représenter des scènes sociales de l'époque. Les nuances et les ombres étaient obtenues à l'aide de traits parallèles.

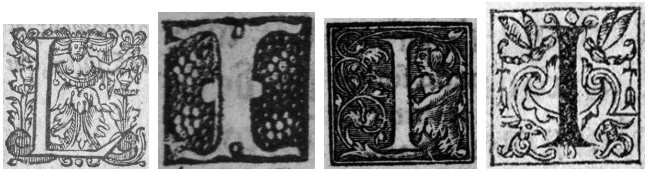


FIGURE 7.7 – Examples de lettrines

L'étude des lettrines fait l'objet de travaux au centre historique CESR de Tours qui en possède un fond documentaire. Les lettrines représentent une source d'information exploitée par les historiens pour situer les documents dans le temps, ou encore pour étudier les scènes sociales des fonds figuratifs. En effet, les tampons étant utilisés plusieurs fois, leur usure permet de situer les documents les uns par rapport aux autres. De plus, ils portent souvent des caractéristiques propres à celui qui les a sculptés.

La mise en place d'outils d'analyse d'images adaptés permet d'envisager des approches automatisées de recherche par le contenu, ou encore de navigation dans un document, lettrine par lettrine.

La sémantique d'une image de lettrine est décrite par un thesaurus issu de travaux historiens [236], qui respecte à la fois le standard TEI (Text Encoding Initiative) [202] - standard de représentation des textes - et la classification Iconclass [267] - système standard de classification des documents d'une collection par sujets. La description des lettrines y est décomposée en quatre couches, chaque couche fournit une information spécifique (voir Figure 7.8) :

Lettre (Letter) : positionnée au centre de l'image, la couche lettre se caractérise en particulier la lettre qu'elle contient, sa couleur (noire ou blanche), l'alphabet (latin, grec, hébreu, ..) et la police utilisée (romain, gothique).

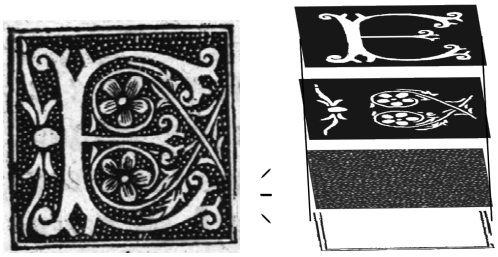
Motif (Pattern) : il est constitué des formes ornementales qui peuvent être décoratives (principalement des motifs) ou figuratives (personnages, bâtiments, arbres ...) ;

Fond (Background) : le fond peut être uniforme (noir ou blanc).

Cadre (Frame) : le cadre correspond aux bords du tampon typographique. Il peut être composé de zéro, un ou deux traits.

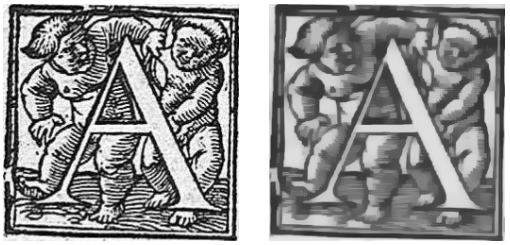
Les images de lettrines sont des images particulièrement difficiles à traiter. En effet, ce sont des images qui se sont détériorées avec le temps - jaunissement du papier, pages abîmées, tâchées - et les techniques développées doivent être robustes à ces détériorations. Par ailleurs, il s'agit d'images binaires à base de traits sur lesquelles les approches texture classiques d'analyse d'images ne peuvent pas s'appliquer.

C'est pourquoi une approche spécifique d'extraction de formes à partir d'images à base de traits a été proposée dans [212] pour extraire des régions correspondant à des formes constitutives d'images de lettrines. En effet, la lettre, ou encore des éléments de scènes naturelles des lettrines à fond figuratif, correspondent à des formes de l'image.



(a) Image initiale (c) Différentes couches

FIGURE 7.8 – Décomposition en couches des historiens



(a) Image d'origine (b) Couche forme

FIGURE 7.9 – Décomposition en couches d'une lettrine

Ces régions sont obtenues en trois étapes :

1. Tout d'abord, l'image est décomposée en plusieurs couches, dont la *couche forme* contenant les formes constitutives de l'image, selon une décomposition décrite dans [219], résultat d'une série de projections [228]. La géométrie d'une image se retrouve alors principalement sur sa couche forme.
2. Ensuite, des composantes connexes de la couche forme sont segmentées à l'aide d'une loi de Zipf [248], particulièrement adaptée par sa robustesse aux variations de niveaux de gris, et par son indépendance à la couleur des composantes.
3. Enfin, ces composantes sont extraites de l'image, formant ainsi des régions. Seules les régions dont l'aire est supérieure à 1% de l'image sont retenues, les autres régions étant considérées moins pertinentes.

Ontologie des lettrines

Nous avons défini une représentation ontologique des lettrines en quatre étapes :

Construction d'une ontologie métier des lettrines : Cette ontologie est une transcription du thesaurus qui décrit la connaissance des historiens. Un fond de 4288

images de lettrines, annotées à la main par les historiens, est utilisé pour peupler la partie A-box de cette ontologie.

Construction d'un ontologie des régions forme des lettrines : Cette ontologie décrit les régions forme d'une image de lettrine. Elle est peuplée par 589 régions extraites de 100 images de lettrines.

Enrichissement de l'ontologie des régions. Nous avons enrichi l'ontologie des régions avec un mécanisme de partitionnement de l'image et des relations spatiales, dans le but de pouvoir localiser chacune des régions selon qu'elle se situe au centre de l'image, ou sur un bord.

Combinaison des deux ontologies, et règles d'inférences. Enfin, nous avons combiné l'ontologie métier et l'ontologie enrichie des régions en une seule *ontologie des lettrines*. Puis nous avons ajouté des descriptions logiques de propriétés permettant d'identifier certaines régions comme la lettre de la lettrine, ou encore une partie d'un personnage du fond de la lettrine.

Nous avons utilisé Protégé 3.4.4 [237] pour construire ces ontologies, et l'outil Jena [235] pour les peupler.

Construction de l'ontologie métier des lettrines La T-box de l'ontologie métier est construite à partir du thesaurus défini par les historiens (cf figure 7.10). Un concept - vocabulaire de la logique de description - ou classe - vocabulaire des ontologies - est défini pour chaque couche sémantique d'une lettrine (*Letter*, *Background*, *Pattern*, *Frame*) ; ainsi que pour spécifier l'alphabet et la police de la lettre (*Background*, *TypeFont*). La classe principale *Lettrine* est reliée à ces concepts par des propriétés :

- *Lettrine* **hasFrame** *Frame*
- *Lettrine* **hasBackground** *Background*
- *Lettrine* **hasPattern** *Pattern*
- *Lettrine* **hasLetter** *Letter*
  - *Lettrine* **hasIdentificationLetter** *IdentificationLetter*
  - *Lettrine* **hasColor** *ColorLetter*
  - *Lettrine* **hasAlphabet** *Alphabet*
  - *Lettrine* **hasTypeFont** *TypeFont*

Construction de l'ontologie des régions forme des lettrines Un concept *Image* permet de représenter chaque image de lettrine. Les concepts *Region* et *RegionSet* représentent les régions, et sont reliés à une lettrine par les propriétés suivantes :

- *Image* **hasRegionSet** *RegionSet*
- *RegionSet* **hasRegion** *Region*

Les informations globales relatives à l'image - unité de mesure, système de référencement utilisé - sont également introduites par des propriétés :

- *Image* **hasImageMeasurementUnit** *MeasurementUnit*

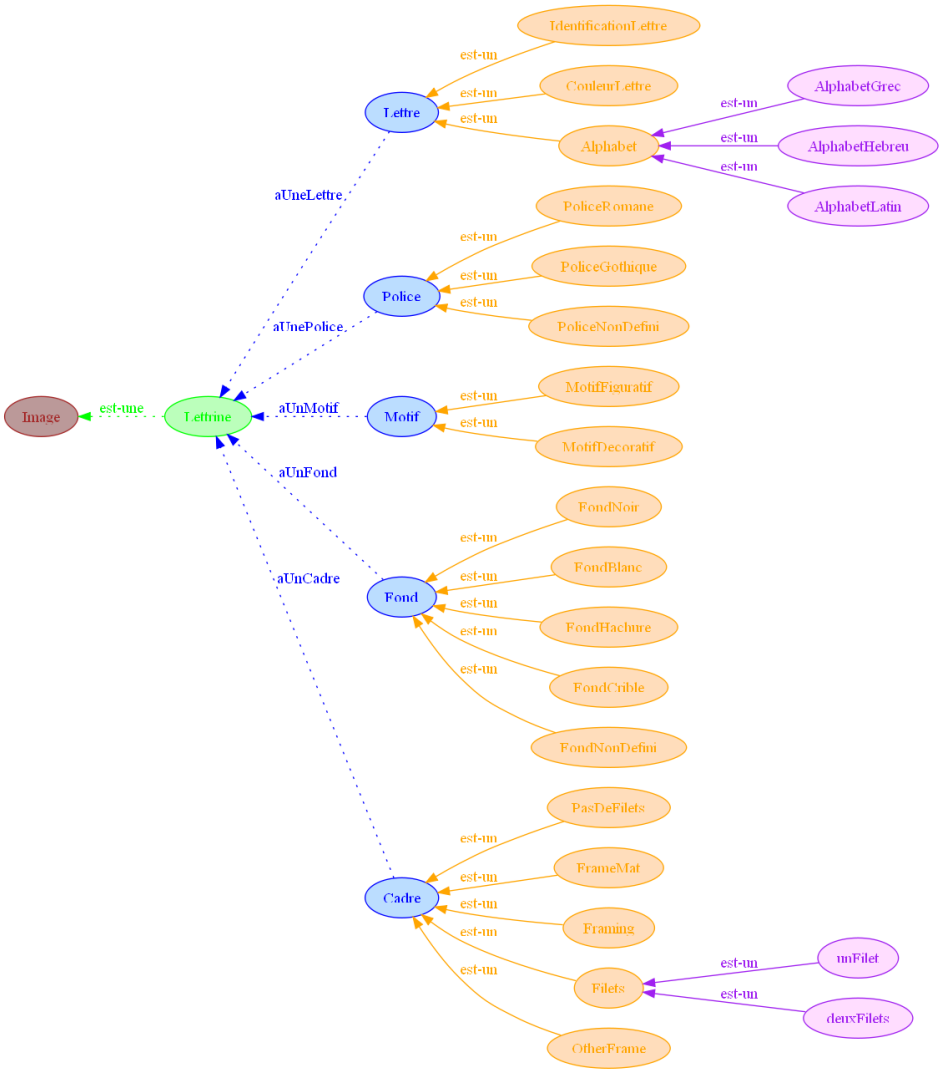


FIGURE 7.10 – Concepts et propriétés de l'ontologie métier d'une lettrine

– *Image* **hasImageReferenceSystem** *ImageReferenceSystem*

Enfin, nous avons sélectionné certaines caractéristiques statistiques des régions, représentées par des propriétés :

– *Region* ***hasX*** et *Region* ***hasY*** : indique les coordonnées du centre de gravité de la région, permettant ainsi de la situer dans l'image.
– *Region* ***hasAire*** : indique l'aire de la région, utilisée pour trier les régions selon leur taille.
– *Region* ***hasExcentricite*** : indique l'excentricité *Ecc* de la région, qui donne une indication de forme. L'excentricité est définie comme le rapport entre le rayon minimal r_m et le rayon maximal r_M de l'ellipse minimale englobant la région [250] :

$$Ecc = \frac{r_M - r_m}{r_M + r_m} \tag{7.1}$$

– *Region* ***hasGreyMean*** et *Region* ***hasGreySTD*** : indique la moyenne en niveaux de gris, ainsi que leur déviation, permettant d'estimer la courleur de la région, ainsi que sa régularité.
– *Region* ***hasEuler*** : indique le nombre d'Euler E_n de la région, pour en estimer la compacité. Le nombre d'Euler introduit dans [250], permet d'estimer le plus généralement la compacité d'un ensemble de régions. Pour une seule région, il se réduit à $E_n = 1 - H$, où H représente le nombre de trous dans la région.

Enrichissement de l'ontologie des régions à l'aide de relations spatiales Nous avons enrichi l'ontologie des régions avec une ontologie standardisée des relations spatiales. L'information spatiale nous intéresse pour pouvoir préciser si chaque région se positionne au centre de l'image, ou sur un bord. Nous avons mis en place un mécanisme de partitionnement d'une image de lettrine. Plus précisément, chaque image est divisée en neuf partitions (voir figure 7.11), une partition correspond au centre de l'image, 4 partitions correspondent aux différents bords, et les 4 dernières aux coins de l'image. Les partitions sont associées à l'image à l'aide d'un concept *Partition* et de la propriété suivante :

– *Image* ***hasPartition*** *Partition*

Les concepts *Region* et *Partition* héritent de la classe abstraite *Surface* du standard SFS de l'OGC (cf Figure 7.5), les huit relations spatiales entre deux objets spatiaux définis par le standard RCC8 (cf Figure 7.6) s'y définissent pas des propriétés.

Cependant, les relations spatiales entre une région et chacune des neuf partitions peuvent également se calculer lors de la phase d'extraction des régions, chaque relation pouvant se caractériser par un booléen pour chaque région et chaque partition. Ainsi, nous pouvons déterminer si une région est située au centre à l'aide de la relation spatiale contains entre cette région et la partition centrale :

– *Region* ***contains*** *Part*

1	2	3
4	5	6
7	8	9

FIGURE 7.11 – Partitionnement d'une image utilisée pour localiser les régions.

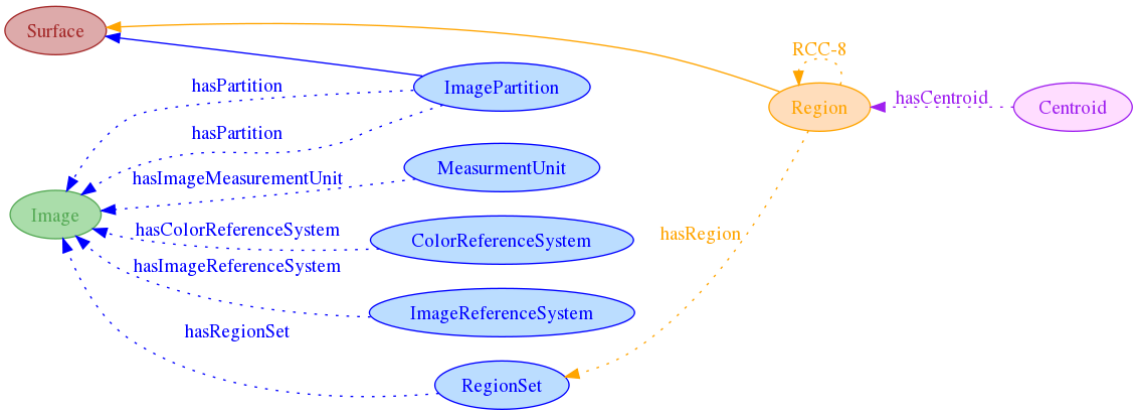


FIGURE 7.12 – Concepts et propriétés de l'ontologie des régions forme d'une lettrine

La fichure 7.12 illustre les concepts et propriétés de cette ontologie des régions forme d'une lettrine ainsi enrichie.

Combinaison des deux ontologies, et règles d'inférence Nous avons ensuite combiné l'*ontologie métier* et l'*ontologie des régions enrichie* en une seule *ontologie des lettrines*. En effet, ces deux ontologies portent chacune une information pour une image de lettrine. Techniquement, la T-box de cette ontologie combinée s'obtient en liant le concept *Lettrine* de l'ontologie métier, et le concept *Image* de l'ontologie des régions. Nous avons utilisé pour cela le constructeur OWL :EQUIVALENTCLASS fourni dans la première version d'OWL. Le peuplement de la A-Box a quant à lui été réalisé à l'aide du constructeur OWL :SAMEAS.

Enfin, nous avons enrichi l'ontologie des lettrines avec de nouvelles propriétés des régions, les propriétés *isLetter* et *isBody* qui intègrent expertise du domaine et expertise de traitement d'images. Il s'agit de propriétés calculées, définies par des règles logiques, permettant d'annoter sémantiquement certaines régions d'une lettrine extraites de la couche forme (cf Figure 7.13) :

Règle pour la propriété *isLetter* **:** indique qu'une région est identifiée comme la lettre

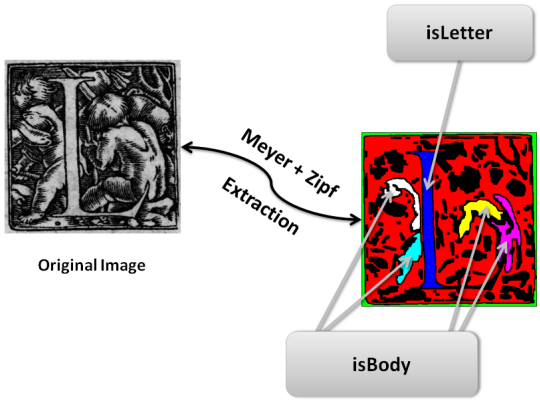


FIGURE 7.13 – Extraction de formes d'une lettrine

d'une lettrine.

(isLetter) *région d'aire maximale*
située au centre de l'image,
et contenant peu de trous.

Plus précisément, cette règle est une combinaison des trois propriétés suivantes qui utilisent des informations de l'ontologie des régions enrichies :

1. *Région située au centre de l'image :* cette propriété se teste à l'aide des relations spatiales. Il s'agit de vérifier que la région est contenue dans la partition centrale, et ne chevauche par les régions du bord.
2. *Region avec peu de trous :* le nombre d'Euler, associé à chaque région dans l'ontologie des régions, est ici utilisé pour tester cette propriété. Après quelques expérimentations, nous avons positionner le nombre d'Euler entre -2 et +2 pour caractériser une région sans trous.
3. *Région d'aire maximale :* dans l'ontologie des régions, l'aire est indiquée pour chaque région. Parmi toutes les régions vérifiant les deux propriétés précédentes, celle d'aire maximale est retenue pour être annotée par la propriété *isLetter*.

Règle pour la propriété *isBody* **:** indique qu'une région est identifiée comme la partie d'un personnage du fond d'une lettrine.

(isBody) *région d'une lettrine figurative*
située au centre de l'image,
sans trous, de couleur gris-clair,
et qui n'est pas une lettre.

Plus précisément, cette règle est une combinaison des cinq propriétés suivantes qui utilisent des informations de l'ontologie des régions enrichies, et de l'ontologie métier :

1. *Région d'une lettrine figurative :* il s'agit là d'une propriété de l'ontologie métier.
2. *Région située au centre de l'image :* comme pour la propriété *isLetter*, cette propriété se teste à l'aide des relations spatiales de l'ontologie des régions enrichie.
3. *Region avec peu de trous :* comme pour la propriété *isLetter*, il s'agit de tester le nombre d'Euler associé à chaque région dans l'ontologie des régions.
4. *Région de couleur gris-clair :* la caractéristique colorimétrique d'une région, présente dans l'ontologie des régions, doit être supérieure à 90 pour indiquer une couleur gris-clair.
5. *Région qui n'est pas une lettre :* parmis toutes les régions vérifiant les propriétés précédentes, seules celles qui ne vérifient pas la propriété *isLetter* sont annotées.

Techniquement, l'outil SWRLTab fourni avec Protégé 3.4.4 est un environnement de développement pour l'édition et l'inférence de règles. Le plugin SWRLJessTab de cet outil propose une version améliorée de l'algorithme Rete d'inférence de règles.

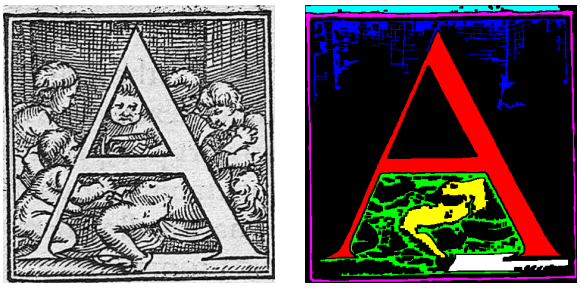
La validation des régions annotées est jusqu'à présent un processus manuel. C'est pourquoi nous avons sélectionné aléatoirement un sous-ensemble de 45 images de lettrines sur lequel appliquer les règles. Cet ensemble est constitué de 27 lettrines à fond décoratif, 18 lettrines à fond figuratif, à partir desquelles 112 régions au total ont été extraites. Les règles pour les propriétés *isLetter* et *isBody* ont été appliquées sur ce sous-ensemble de 45 images. Les résultats obtenus sont encourageants :

Règle pour la propriété *isLetter* **:** Après vérification manuelle, 39 régions s'avèrent être correctement annotées, et 6 ne le sont pas, d'où un taux de reconnaissance de 86,6%.

Il serait ici possible de comparer la forme de la région avec celle attendue de la lettre, information présente dans l'ontologie métier. Une telle extension permettrait de détecter les cas où la lettre se décompose en plusieurs régions, et ainsi réitérer la règle pour la propriété *isLetter* jusqu'à ce que toutes les régions de la lettre soient détectées.

Règle pour la propriété *isBody* **:** Appliquée sur ce sous-ensemble de 45 images de lettrines, cette règle permet d'annoter correctement des régions de 17 images sur 18 images figuratives, d'où un taux d'erreur inférieur à 2%.

La figure 7.14 représente la lettrine figurative dont les régions ne sont pas correctement annotées. On peut y observer que la région jaune correspondant à une



(a) Image de lettrine (b) Régions extraites

FIGURE 7.14 – L'image où des régions ne sont pas correctement identifiées comme partie de personnage par *isBody*

partie d'un personnage n'est pas annotée, son nombre d'Euler est trop grand. En revanche, la région blanche est annotée alors qu'elle ne correspond pas à une partie d'un personnage. L'utilisation d'un partitionnement plus complexe de l'image, ou encore d'autres descripteurs statistiques des régions, pourrait permettre d'annoter correctement les régions de ce type.

7.2.4 Conclusion

Ces travaux autour des lettrines sont des travaux récents, plusieurs perspectives sont d'ores et déjà envisagées, voire amorcées. L'amélioration des deux règles existantes est une première perspective : pour la première règle, possibilité de comparer la région lettre avec la forme attendue ; pour la seconde règle, mise en place d'un partitionnement plus sophistiqué, et utilisation de nouveaux descripteurs des régions. Nous souhaiterions également intégrer de nouvelles règles pour les régions existantes, ou pour de nouveaux types de régions. Dans des travaux en cours, des régions extraites d'une autre couche - la *couche texture* - de la décomposition en couche d'une image de lettrine permettent ainsi de décrire des zones de traits qui semblent pertinentes. Enfin, l'intégration d'une vérité terrain des régions d'une lettrine nous semble indispensable pour pouvoir vérifier automatiquement - et non manuellement - les règles mises en place, et ainsi les appliquer sur une plus grande quantité de régions et de lettrines.

La représentation ontologique des lettrines présentée dans cette section a pour principal avantage de pouvoir combiner dans un cadre unifié des informations diverses décrivant les lettrines, à savoir :

- une *annotation sémantique* décrivant le contenu des lettrines,
- des *régions forme* constituant les lettrines,
- des *descripteurs statistiques* décrivant les régions,
- une *information spatiale* entre ces régions et un partitionnement des lettrines,
- une *annotation sémantique* de certaines régions.

Il s'agit d'informations de sources variées, obtenues :

à partir d'une information haut-niveau - un thésaurus défini par des historiens,

à partir d'une information bas-niveau - une extraction de région formes et de caractéristiques numériques les décrivant,

par un mécanisme d'inférence logique qui peut s'appuyer sur les relations spatiales. Deux règles logiques ont ainsi été définies.

Etablir un lien entre une information bas-niveau et une information haut-niveau décrivant des images est une problématique importante dans le domaine de l'analyse d'images, identifiée sous le terme de *fossé sémantique*. L'approche proposée est ainsi une contribution à une réduction de ce fossé sémantique pour des images de lettrines, images fortement structurées issues d'un domaine porteur d'une connaissance métier importante. Nous avons ainsi pu identifier plusieurs caractéristiques propres au mécanisme d'inférence proposé :

Inférence à partir de caractéristiques bas-niveau. La première règle *isLetter* permet une annotation de certaines régions à partir de leurs caractéristiques bas-niveau.

Inférence combinant les deux types d'information. La seconde règle *isBody* combine des caractéristiques bas-niveau des régions, mais aussi une information haut-niveau. En effet, seules les régions d'une lettrine figurative peuvent être annotées.

Inférence itérative. La seconde règle *isBody* s'appuie sur le résultat de la première. En effet, seules les régions qui n'ont pas préalablement été annotées comme étant une lettre pourront être annotées.

Il s'agit là de caractéristiques propres à l'approche proposée, que nous souhaiterions pouvoir étendre à des images graphiques d'un autre domaine. Des travaux autour d'images de bandes dessinées sont envisagés.

Troisième partie

Conclusion

Une synthèse de mes travaux de recherche a été présentée dans ce document. Ces recherches posent la structure de treillis en premier plan, avec des contributions sur des aspects structurels et algorithmiques, et sur quelques usages pour des données images. Mes travaux de recherche se situent à la fois sur un plan fondamental - avec des contributions d'ordre théorique où des propriétés et algorithmes sont établis à l'aide de preuve - mais également sur un plan applicatif - avec des propositions de nouvelles méthodes validées par des études experimentales. C'est la raison pour laquelle ce manuscrit est organisé en deux parties dont je rappelle brièvement les points forts.

Partie I : Structure de treillis : concepts de base et algorithmes

L'objectif principal de la synthèse proposée dans la première partie est de fournir les fondamentaux nécessaires à une manipulation efficace d'un treillis et de ses composants bijectifs. Cette synthèse introduit le treillis de fermés de façon générique pour un système de fermeture quelconque, autour duquel gravite à la fois la structure de treillis, mais aussi celle de treillis des concepts, de contexte, de treillis de Galois, et de système implicatif. Il apparaît en effet que les systèmes de fermeture jouent un rôle majeur dans la théorie des treillis du fait du lien bijectif qui les unit, permettant ainsi d'étendre le champ applicatif des treillis à celui très riche des systèmes de fermeture que l'on retrouve dans de nombreux domaines. Les principales contributions présentées dans cette partie peuvent se résumer en deux points :

1. Etude de la *base canonique directe d'un treillis*, d'un point de vue structurel et algorithmique [16][11]
2. Développement de la bibliothèque Lattice de manipulation d'un système de fermeture qui intègre de *nouveaux algorithmes* que nous avons développés.

En terme de perspectives, il me semble tout d'abord important de continuer à maintenir un *lien entre recherche fondamentale et recherche appliquée autour de la structure de treillis*, avec l'algorithmique comme élément central. La synthèse proposée dans cette partie, ainsi que la bibliothèque Lattice, sont une contribution en ce sens.

Par ailleurs, les travaux autour de la *base canonique directe sont prometteurs*. La question d'une caractérisation mutuelle entre les deux bases - la canonique et la canonique directe - me semble pertinente. Une telle caractérisation aurait un impact sur une utilisation conjointe de ces deux bases dans un cadre applicatif. En effet, il serait alors possible de cumuler les avantages algorithmiques de la base canonique directe, et le pouvoir de résumé des données de la base canonique.

Une autre perspective intéressante concerne les règles d'implications portées par le ODGraph, structure de graphe similaire à celle du graphe de dépendance, mais de taille polynomiale en celle du treillis. Des travaux actuels conjecturent que ce graphe encode une base non pas canonique directe, mais *canonique directe-ordonnée*. C'est-à-dire qu'il existe un ordre de rangement des règles de telle sorte à ce qu'un calcul de fermeture

soit possible en une seule itération. Si cette propriété s'avère vérifiée, il s'agirait alors d'une nouvelle base similaire à la base canonique directe, mais de taille polynomiale en celle du treillis.

Partie II : Quelques usages pour des données images

Cette seconde partie est consacrée à quelques usages des treillis en informatique pour traiter des données, et en particulier des images. Nos travaux ont porté à la fois sur l'exploration ou la fouille de données dans un objectif de classification, et sur la représentation de relations spatiales entre primitives issues d'une image, en particulier par une ontologie. Un travail conséquent de synthèse des méthodes issues de l'AFC existant en fouille de données et en représentation ontologique des connaissances y est présenté, l'objectif étant de positonner les différentes approches les unes par rapport aux autres. Les principales contributions présentées dans cette partie peuvent se résumer en trois points :

1. Mise en place de la méthode NAVIGALA de reconnaissance d'image de symboles par navigation dans un treillis des concepts [34], implémentée dans un logiciel du même nom.
2. Etude des *treillis dichotomiques*, avec notament un algorithme de *discrétisation locale* de données numériques [18]
3. Représentation de l'information spatiale entre primitives d'une image, avec la mise en place d'une *signature structurelle d'images de symboles* [19], et d'une *ontologie des lettrines*.

En terme de perspectives, la poursuite des travaux engagés autour d'une *méthode de classification par navigation* me semble prometteuse. L'étude des treillis dichotomiques laisse envisager des perspectives intéressantes pour définir une structure hybride de classification, intermédiaire entre treillis et arbre, dans le but à la fois d'améliorer les performances de l'arbre et de limiter la trop grande taille du treillis, tout en garantissant la lisibilité de ces structures. Ces travaux font l'objet d'une thèse en cours. Par ailleurs, le mécanisme de discrétisation locale, conjointement avec la génération à la demande, laisse envisager des possibilités d'intégration dynamique et interactive de nouveaux attributs au processus de classification.

Une autre direction de recherche concerne les travaux amorcés autour des *représentations spatiales et ontologiques d'images*. Alors que l'information spatiale permet une description structurelle d'images fortement structurées, les mécanismes de raisonnement propres aux ontologies permettent de déduire de la connaissance implicite à partir de la connaissance explicitement formalisée. Des travaux autour d'images de bandes dessinées sont envisagés.

Bibliographie

Publications liées à ce mémoire

[1] M. AlHajj, K. Bertet, J. Gay, and J.-M. Ogier. Using the concept lattice for graphic understanding. In *Procedings of the fifth IAPR International Workshop on Graphics Recognition (GREC'03)*, Barcelona, Spain, July 2003.

[2] K. Bertet. *Sur quelques aspects algorithmiques et structurels des treillis.* PhD thesis, Univeristé Paris 7, 1998.

[3] K. Bertet. Representation of a lattice associated to an implicationnal Moore family by the direct-optimal implicationnal system. In *ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, Maryland, Jan. 2003.

[4] K. Bertet. Some algorithmical aspects using the canonical direct implicational basis. In S. Ben-Yahia and E. Mephu-Nguifo, editors, *Proceedings of fourth International Conference on Concept Lattices and their Applications (CLA'06)*, Yasmine Hammamet, Tunisia, Oct. 30 - Nov. 1 2006.

[5] K. Bertet and N. Caspard. Doubling convex sets in lattices : characterisations. In *Third International Conference on Orders, Algorithms and Applications (ICOOA'99)*, Montpellier, France, Aout 1999.

[6] K. Bertet and N. Caspard. Doubling convex sets in lattices : characterisations and recognition algorithms. *Order*, 19(2) :181–207, Sept. 2002.

[7] K. Bertet, C. Chaudet, I. Guérin-Lassous, and L. Viennot. Impact of interferences on bandwidth reservation for ad hoc networks : a first theoretical study. In *Procedings of the IEEE Symposium on Ad-Hoc Wireless Networks (GLOBECOM SAWN'01)*, 2001.

[8] K. Bertet, S. Guillas, and J.M. Ogier. Extensions of bordat's algorithm for attributes. In *Proceedings of Fifth International Conference on Concept Lattices and their Applications (CLA'07)*, pages 38–49, Montpellier, France, Oct. 24-26 2007.

[9] K. Bertet, S. Guillas, and J.M. Ogier. On the joint use of two implicational bases. In *Proceedings of Fifth International Conference on Formal Concept Analysis (ICFCA'07)*, pages 23–39, Clermont-Ferrand, France, Feb. 12-16 2007.

[10] K. Bertet, D. Krob, M. Morvan, J-C. Novelli, H. D. Phan, and J-Y. Thibbon. Some algoritmhs for computing plethysms of quasi-symmetric functions in the sense of Malvenuto and Reutenauer. *Communication on Algebra*, 29(9), September 2001.

[11] K. Bertet and B. Monjardet. The multiple facets of the canonical direct unit implicational basis. *Theoretical Computer Science*, 411(22-24) :2155–2166, 2010.

[12] K. Bertet and M. Morvan. Computing the sublattice of a lattice generated by a set of elements. In *Third International Conference on Orders, Algorithms and Applications (ICOOA'99)*, Montpellier, France, Aout 1999.

[13] K. Bertet, M. Morvan, and J. Gustedt. Weak-orders extensions of an order. In *23st Workshop on Graph-Theoretic Concept in Computer Science (WG'97)*, Berlin, Germany, June 1997.

[14] K. Bertet, M. Morvan, and J. Gustedt. Weak-order extensions of an order. *Theoretical Computer Science*, 304 :249–268, 2003.

[15] K. Bertet, M. Morvan, and L. Nourine. Lazy completion of a dag to the smallest lattice. In *Second Internationnal Sympsosium on Knowledge Retrieval, Use and Storage for Efficiency (KRUSE'97)*, Vancouver, Canada, August 1997.

[16] K. Bertet and M. Nebut. Efficient algorithms on the family associated to an implicationnal system. *Discrete Mathematical and Theoretical Computer Science*, 6 :315–338, 2004.

[17] K. Bertet and J.M. Ogier. Graphic recognition : The concept lattice approach. *Lecture Notes in Computer Science, Graphics Recognition - Recent Advances and Perspectives*, 3088, August 2004.

[18] K. Bertet, M. Visani, and N. Girard. Treillis dichotomiques et arbres de décision. *Traitement du Signal (TS)*, 26(5) :407–416, 2009.

[19] M. Coustaty, K. Bertet, M. Visani, and J.M. Ogier. A New Adaptive Structural Signature for Symbol Recognition by Using a Galois Lattice as a Classifier. *IEEE Transactions on Systems, Man, and Cybernetics*, to appear, 2011.

[20] M. Coustaty, S. Guillas, M. Visani, K. Bertet, and J.-M. Ogier. Reconnaissance de symboles à partir d'une signature structurelle flexible et d'un classifieur de type treillis de Galois. *Techniques et Sciences Informatiques (TSI)*, 29(6) :1–26, 2010.

[21] M. Coustaty, S. Guillas, M. Visani, K. Bertet, and J.M. Ogier. Flexible structural signature for symbol recognition using a concept lattice classifier. In *Proceedings of Seventh IAPR International Workshop on Graphics Recognition (GREC'07)*, Curitiba, Brazil, September 20-21 2007.

[22] M. Coustaty, S. Guillas, M. Visani, K. Bertet, and J.M. Ogier. *Graphics Recognition : Recent Advances and New Opportunities, Lecture Notes in Computer Science (LNCS), (special issue post GREC'07)*, volume 5046, chapter On the

joint use of a structural signature and a Galois lattice classifier for symbol recognition, pages 61–70. Springer-Verlag, 2008. Revised and extended version of paper first presented at Seventh IAPR International Workshop on Graphics Recognition (GREC'07).

[23] N. Girard, K. Bertet, and M. Visani. Vers une discrétisation locale pour les treillis dichotomiques. In *Actes XVIèmes Rencontres de la Société Francophone de Classification (SFC'09)*, pages 113–116, Grenoble, France, Septemre 2009.

[24] S. Guillas, K. Bertet, and J.M. Ogier. Concept Lattices : a Tool for Supervised Classification ? In *Proceedings of Sixth IAPR Workshop on Graphics Recognition (GREC'05)*, pages 56–67, Hong Kong, China, August 2005.

[25] S. Guillas, K. Bertet, and J.M. Ogier. Les treillis de Galois : un outil pour la sélection de primitives ? *Traitement du Signal (TS)*, 22(3) :273–291, 2005.

[26] S. Guillas, K. Bertet, and J.M. Ogier. Concept lattice classifier : a first step towards an iterative process of recognition of noised graphic objects. In S. Ben-Yahia and E. Mephu-Nguifo, editors, *Proceedings of fourth International Conference on Concept Lattices and their Applications (CLA'06)*, pages 257–263, Yasmine Hammamet, Tunisia, Oct. 30 - Nov. 1 2006.

[27] S. Guillas, K. Bertet, and J.M. Ogier. A generic description of the concept lattices' classifier : Application to symbol recognition. *Graphics Recognition : Ten Years Review and Future Perspectives (Revised Selected Papers)*, 3926 :47–60, 2006. LNCS, Springer Berlin / Heidelberg. Revised and extended version of paper first presented at Sixth IAPR International Workshop on Graphics Recognition (GREC'05).

[28] S. Guillas, K. Bertet, and J.M. Ogier. Reconnaissance de symboles bruités à l'aide d'un treillis de Galois. In L. Likforman-Sulem, editor, *Neuvième Colloque International Francophone sur l'Ecrit et le Document (CIFED'06)*, pages 85–90, Fribourg, Suisse, 2006. isbn 2-9522067-1-6.

[29] S. Guillas, K. Bertet, and J.M. Ogier. *Concept Lattices and their Application, Lecture Notes in Artificial Intelligence (LNAI), (special issue post CLA'06)*, volume 4923, chapter Towards an iterative classification based on concept lattice, pages 256–262. Springer-Verlag, 2008. Revised and extended version of paper presented at Fourth International Conference on Concept Lattices and their Applications (CLA'2006).

[30] S. Guillas, K. Bertet, and J.M. Ogier. A propos des liens entre arbre de décision et treillis dichotomique. In *Dixième Colloque International Francophone sur l'Ecrit et le Document (CIFED'08)*, Rouen, France, 2008.

[31] S. Guillas, K. Bertet, M. Visani, J-M. Ogier, and N. Girard. Some links between decision tree and dichotomic lattice. In *Proceedings of Sixth International Conference on Concept Lattices and their Applications (CLA'08)*, pages 193–

205, Olomouc, Czech Republic, Oct. 21-23 2008. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-433/.

[32] M. Rusinol, K. Bertet, J.-M. Ogier, and J. Llados. Symbol recognition by using a concept lattice of graphical patterns. In *Proc. of IAPR International Workshop on Graphics Recognition (GREC'09)*, La Rochelle, France, July 2009.

[33] M. Rusinol, K. Bertet, J.-M. Ogier, and J. Llados. *Graphics Recognition : Recent Advances and New Opportunities, Lecture Notes in Computer Science (LNCS), (special issue post GREC'09)*, volume 6020, chapter Symbol recognition by using a concept lattice of graphical patterns. Springer, 2010. Revised and extended version of paper first presented at Seventh IAPR International Workshop on Graphics Recognition (GREC'09).

[34] M. Visani, K. Bertet, and J-M. Ogier. Navigala : an Original Symbol Classifier Based on Navigation through a Galois Lattice. *International Journal on Pattern Recognition and Artificial Intelligence (IJPRAI)*, to appear, 2011.

Mémoire de travaux encadrés

[35] M. Al-Haj. Etude expérimentale d'une technique de reconnaissance d'images détériorées. Master's thesis, Université de La Rochelle, La Rochelle, France, 2003.

[36] S. Bernard. Extraction de primitives structurelles pour la reconnaissance de symboles : Une approche robuste par Transformée de Hough. Master's thesis, Université de La Rochelle, La Rochelle, France, 2006.

[37] R. Bertrand. Génération mutli-threadée de treillis de Galois. Master's thesis, Université de La Rochelle, La Rochelle, France, 2010.

[38] M. Coustaty. Mise en oeuvre de techniques de reconnaissance de symboles. Master's thesis, Université de La Rochelle, La Rochelle, France, 2007.

[39] N. Girard. Etude comparative d'algorithmes de classification supervisée basés sur les treillis de Galois. Master's thesis, Université de La Rochelle, La Rochelle, France, 2006.

[40] S. Guillas. Reconnaissance d'objets graphiques détériorés : approche fondée sur un treillis de Galois. Master's thesis, Université de La Rochelle, La Rochelle, France, 2007.

[41] H. Khene. Algorithme de reconnaissance d'objets détériorés basé sur un treillis de Galois. Master's thesis, Université de La Rochelle, La Rochelle, France, 2004.

[42] T. Le Ngoc. Utilisation de méthodes pour enrichir des données vectorielles simples avec des relations spatiales et des règles. Master's thesis, Université de La Rochelle, La Rochelle, France, 2010.

[43] A. Mercier and Ph. Sachot. Création d'une interface graphique pour un prototype de logiciel de reconnaissance de symboles. Master's thesis, Université de La Rochelle, La Rochelle, France, 2010.

Aspects structurels des treillis

[44] W.W. Armstrong. Dependency structure of data base relationships. In *IFIP Congress*, 1974.

[45] M. Barbut and B. Monjardet. *Ordres et classifications : Algèbre et combinatoire*. Hachette, Paris, 1970. 2 tomes.

[46] G. Birkhoff. On the combination of subalgebras. In *Proc. of the cambridge Philisophical Society*, volume 29, pages 441–464, 1933.

[47] G. Birkhoff. *Lattice theory*. American Mathematical Society, 1st edition, 1940.

[48] G. Birkhoff. *Lattice theory*. American Mathematical Society, 3d edition, 1967.

[49] N. Caspard, B. Leclerc, and B Monjardet. *Ensembles ordonnés finis : concepts, résultats et usages*. Springer-Verlag, 2007.

[50] N. Caspard and B. Monjardet. The lattice of closure systems, closure operators and implicational systems on a finite set : a survey. *Discrete Applied Mathematics*, 127(2) :241–269, 2003.

[51] B.A. Davey and H.A. Priestley. *Introduction to lattices and orders*. Cambridge University Press, 2nd edition, 1991.

[52] A. Day. A simple solution of the word probleme for lattices. *Canad. math. Bull.*, 13 :253–254, 1970.

[53] A. Day. Doubling convex sets in lattices and a genaralized semidistributivity condition. *Order*, 6 :175–180, 1989.

[54] J.P. Doignon and J.C. Falmagne. *Knowledge spaces*. Springer Verlag, Berlin, 1999.

[55] S. Ferré and O. Ridoux. An introduction to logical information systems. *Information Processing & Management*, 40(3) :383–419, 2004.

[56] R. Freeze, J. Jezek, and J.B. Nation. Free lattices. In Providence, editor, *Mathematical survey and monographs. Americal Mathematical Society*, volume 42, 1995.

[57] B. Ganter and S.O. Kuznetsov. Pattern structures and their projections. In *LNCS of International Conference on Conceptual Structures (ICCS'01)*, pages 129–142, 2001.

[58] B. Ganter and R. Wille. *Formal Concept Analysis, Mathematical foundations*. Springer Verlag, Berlin, 1999.

[59] G Grätzer. *General lattice theory*. Birkahäuser-Verlag and Springer, 1st edition, 1978.

[60] J.L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines*, 95 :5–18, 1986.

[61] A. Hörn. On sentences wich are true of direct unions of algebras. *Journal of Symbolic logic*, 16 :14–21, 1951.

[62] E. V. Huntington. Sets of independent postulates for the algebra of logic. *Transactions of the American Mathematical Society*, 5(3) :288–309, 1904.

[63] T. Ibaraki, A. Kogan, and K. Makino. Functional dependencies in Horn theories. *Artificial Intelligence*, 108 :1–30, 1999.

[64] M. Kaytoue. *Traitement de données numériques par analyse formelle de concepts et structures de patrons*. PhD thesis, Université henry Poincaré, Nancy, 2011.

[65] D. Maier. *The Theory of Relational Databases*. Computer Sciences Press, 1983.

[66] G. Markowsky. Some combinatorial aspects on lattice theory. In *Proc. of Houston lattice theory conference*, pages 36–68, 1973.

[67] G. Markowsky. The representation of posets and lattices by sets. *Algebra Universalis*, 11 :173–192, 1980.

[68] B. Monjardet. Arroxian characterizations of latticial federation consensus. *Mathematical Social Sciences*, 20(1) :51–71, 1990.

[69] B. Monjardet. La construction des notions d'ordre et de treillis. In *Journées nationales de l' Association des Professeurs de Mathématiques de l' Enseignement Public (APMEP)*, La Rochelle, France, Oct. 2008.

[70] B. Monjardet and N. Caspard. On a dependence relation in finite lattices. In *Proc. of an International Symposium on Graphs and Combinatorics*, pages 497–505, Amsterdam, The Netherlands, 1997. Elsevier Science Publishers.

[71] E.H. Moore. On a form of general analysis with applications to linear differential and integral equations. In *Atti del IV Congresso Internazionale dei Matematici*, pages 98–114, Roma, 1909.

[72] J. B. Nation. An approach to lattice varieties of finite height. *Algebra Universalis*, 27(4) :521–543, 1990.

[73] N. Pasquier. *Data mining : Algorithmes d'extraction et de réduction des règles d'association dans les bases de données*. PhD thesis, Université of Clermont-Ferrand II, 2000.

[74] C.S. Peirce. On the algebra on logic. *American Journal of mathematics*, 3 :15–57, 1880.

[75] G. Polaillon. *Organisation et interprétation par les treillis de Galois de données de type multivalué , intervalle ou histogramme*. PhD thesis, Paris IX-Dauphine, 1998.

[76] O. Öre. Galois connexion. *Transactions of the American Mathematicl Society*, 55, 1944.

[77] A. Rusch and R. Wille. Knowledge spaces and formal concept analysis. In H.H. Bock and W. Polasek, editors, *Data Analysis and Information Systems*, pages 427–436, Berlin, 1995. Springer Verlag.

[78] L. Santocanale. A duality for finite lattices. http://hal.archives-ouvertes.fr/hal-00432113, 2009.

[79] G. Szasz. *Introduction à la théorie des treillis*. Dunod, Paris, 1971.

[80] R. Taouil and Y. Bastide. Computing proper implications. In *Ninth International Conference on Conceptual Structures (ICCS'01)*, Stanford, USA, 2002.

[81] M. Wild. A theory of finite closure spaces based on implications. *Advances in Mathematics*, 108(1) :118–139, 1994.

[82] R. Wille. Restructuring lattice theory : an approach based on hierarchies of concepts. *Ordered sets*, pages 445–470, 1982. I. Rival (ed.), Dordrecht-Boston, Reidel.

[83] R. Wille. Subdirect decomposition of concept lattices. *Algebra universalis*, 17 :275–287, 1983.

Aspects algorithmiques des treillis

[84] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J.B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. of 20th International Conference on Very Large Data Bases (VLBD'94)*, pages 487–499. Morgan Kaufmann, 1994.

[85] M.A. Babin and S.O. Kuznetsov. Recognizing pseudo-intents is coNP-complete. In M. Kryszkiewicz and S.O. Kuznetsov, editors, *Proc. of the 7th International Conference on Concept Lattices and their Applications (CLA'11)*, pages 294–301, 2011.

[86] J.P. Bordat. Calcul pratique du treillis de Galois d'une correspondance. *Math. Sci. Hum.*, 96 :31–47, 1986.

[87] M. Chein. Algorithme de recherche des sous-matrices premières d'une matrice. *Bulletin Mathématique de la Sociologie Scientifique de la R.S. de Roumanie*, 1969.

[88] A. Le Floch, C. Fisette, R. Missaoui, P. Valtchev, and R. Godin. Jen : un algorithme efficace de construction de générateurs minimaux pour l'identification de règles d'association. *Nouvelles Technologies de l'Information (numéro spécial)*, 1(1) :135–146, 2003.

[89] M.L. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21 :618–628, 1996.

[90] Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *Software - Practice and Experience*, 1999. http://www.graphviz.org/.

[91] B. Ganter. Two basic algorithms in concept analysis. *Technische Hochschule Darmstadt (Preprint 831)*, 1984.

[92] R. Gödin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on Galois (concept) lattices. *Computational Intelligence*, 11(6) :246–267, 1995.

[93] A. Gely. A generic algorithm for generating closed sets of binary relation. *Third International Conference on Formal Concept Analysis (ICFCA 2005)*, pages 223–234, Feb. 2005.

[94] A. Goralcikova and V. Koubek. *Mathematical Foundations of Computer Science, Lecture Notes in Computer Science*, volume 74, chapter A reduct and closure algorithm for graphs, pages 301–307. Springer, 1979.

[95] M. Habib, R. Medina, L. Nourine, and G. Steiner. A survey on generating ideals for posets and applications. Research Report 95-033, LIRMM, Montpellier, July 1995.

[96] D.J. Kavvadias and E.C. Stravropoulos. Monotone boolean dualization is in co-np[$log^2 n$]. *Information Processing Letter*, 85 :1–6, 2003.

[97] P. Krajca and V. Vychodil. Comparison of data structures for computing formal concepts. In Springer-Verlag, editor, *Proc. of the 6th International Conference on Modeling Decisions for Artificial Intelligence (MDAI'09)*, pages 114–125. Springer-Verlag LNAI, 2009.

[98] S. Kuznetsov and S. Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theorical Artificial Intelligence*, 14(2-3) :189–216, 2002.

[99] S.A. Kuznetsov. On the intractability of computing the Duquenne-Guigues base. *Journal of Universal Computer Sciences*, 8(10) :237–243, 2004.

[100] E. Lawler, J.K. Lenstra, and A.H.G. Rinnoy Kan. Generating all maximal independant sets : Np-hardness and polynomial time algorithms. *SIAM Journal on Computing*, 9 :558–565, 1980.

[101] C. Linding. Fast concept analysis. In *Working with Conceptual Structures – Contributions to the International Conference on Computational Science (ICCS'00)*, pages 152–161, Germany, 2000.

[102] H.M. MacNeille. Partially ordered sets. *Trans. Amer. Soc.*, 42 :416–460, 1937.

[103] H. Mannila and K.J. Räihä. *The design of relational databases*. Addison-Wesley, 1992.

[104] M. Morvan and L. Nourine. Simplicial elimination scheme, extremal lattices and maximal antichains lattice. *Order*, 13(2) :159–173, 1996.

[105] E. Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2), 1978.

[106] L. Nourine. Une structuration algorithmique de la théorie des treillis, 2000. Habilitation. Université of Montpellier I.

[107] L. Nourine and O. Raynaud. A fast algorithm for building lattices. *Information Processing Letters*, 71 :199–204, 1999.

[108] S.A. Obiedkov and V. Duquenne. Attribute-incremental construction of the canonical implication basis. *Annals of Mathematics and Artificial Intelligence*, 49(1-4) :77–99, 2007.

[109] JL. Pfältz and CM. Taylor. Scientific discovery through iterative transformations of concept lattices. In *Workhop on Discrete Applied Mathematics, in conjonction with the second SIAM International Conference on Data-Mining (ICDM'02)*, pages 65–74, 2002.

[110] U. Priss. Fca software interoperability. In *Proc. of Sixth International Conference on Concept Lattices and their Applications (CLA'08)*, pages 193–205, Olomouc, Czech Republic, Oct. 21-23 2008. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-433/.

[111] Y. Renaud. *Quelques aspects algorithmiques sur les systèmes de fermeture*. PhD thesis, Univeristé Clermont-Ferrand II, 2008.

[112] R.C. Shock. Computing the minimal cover of functional dependencies. *Information Processing Letters*, 3(22) :157–159, 1986.

[113] P. Valtchev, R. Godin, R. Missaoui, M. Huchard, and A. Napoli. Galois lattice interactive constructor, 2006. http://www.iro.umontreal.ca/galicia. Dernière visite : Oct. 2011.

[114] F. Vogt and R. Wille. Toscana - a graphical tool for analyzing and exploring data. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing. Lecture Notes in Computer Sciences*, pages 226–233, 1994. http://toscanaj.sourceforge.net.

[115] M. Wild. Implicational bases for finite closure system, 1989. preprint.

[116] M. Wild. Computations with finite closure systems and implications. In *Proc. of the 1st Annual International Conference on Computing and Combinatorics (ICC'95)*, volume 959 of *LNCS*, pages 111–120. Springer, 1995.

[117] S. A. Yevtushenko. System of data analysis concept explorer(in russian). In *Proc. of 7th national conference on Artificial Intelligence (KII'00)*, pages 127–134, Russia, 2000. http://sourceforge.net/projects/conexp,dernièrevisite:Oct.2011.

Extraction des connaissances et AFC

[118] P. Adriaans and D. Zantigue. *Data-mining*. Addison-Wesley, 1996.

[119] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In ACM Press, editor, *Proc. of the ACM- International Conference on Management of Data (SIGMOD'93)*, pages 207–216, May 1993.

[120] M. Antonie and O. Zaiane. Classifying text documents by associating terms with text categories. In *Proc. of the Thirteenth Austral-Asian Database Conference (ADC'02)*, Melbourne, Australia, 2002.

[121] M. Antonie and O. Zaiane. Text document categorization by term association. In *Proc. of IEEE International Conference on Data Mining (ICDM'02)*, pages 19–26, Maebashi City, Japan, 2002. IEEE Computer Society.

[122] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Pascal : un algorithme d'extraction des motifs fréquents. *Technique et science informatiques*, 21(1) :65–75, 2002.

[123] R.J. Bayardo. Efficient mining long pattern from databases. In ACM Press, editor, *Proc. of the ACM International Conference on Management of Data (SIGMOD'98)*, pages 85–93, June 1998.

[124] S.O. Belkasim, M. Shridar, and M. Ahmadi. Pattern recognition with moment invariants : a comparative study and new results. *Pattern Recognition*, 24 :1117–1138, 1991.

[125] R. Belohlavek. *Fuzzy Relational Systems : Foundations and Principles*. Kluwer Academic/Plenum Press, New York, 2002. ISBN 0-306-46777-1.

[126] R. Belohlavek, B. De Baets, J. Outrata, and V. Vychodil. Inducing decision trees via concept lattices. In *Proc. of the fifth International Conference on Concept Lattices and their Applications (CLA'07)*, pages 38–49, Montpellier, France, Oct. 24-26 2007.

[127] M. Berry and G. Lino. *Data-mining*. Addison-Wesley, 1996.

[128] I. Bouzouita, S. Elloumi, and S. Ben-Yahia. GARCm : Generic Association Rules based Classifier Multi-parameterizable. In S. Ben-Yahia and E. Mephu-Nguifo, editors, *Proc. of the Fourth International Conference on Concept Lattices and their Applications (CLA'06)*, pages 115–125, Yasmine Hammamet, Tunisia, 2006.

[129] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Wadsworth Inc., Belmont, California, 1984.

[130] S. Brin, R. Motwani, J.D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In ACM Press, editor, *Proc. of the*

ACM- International Conference on Management of Data (SIGMOD'97), pages 255–264, May 1997.

[131] F. Brucker and J.-P. Barthélemy. *Eléments de classification*. Hermes publishing, 2007.

[132] F. Brucker and A. Gély. Parsimonious cluster systems. *Advances in Data Analysis and Classification*, 9 :189204, 2009.

[133] C. Carpineto and G. Romano. Galois : An order-theoretic approach to conceptual clustering. In *Proc. of the International Conference on Machine Learning (ICML'93)*, pages 33–40, Amherst, July 1993.

[134] C. Carpineto and G. Romano. *Concept Data Analysis : Theory and Applications*. John Wiley & Sons, 2004.

[135] A. Chhabra and I. Phillips. The second international graphics recognition contest - raster to vector conversion : A report. In *Graphics recognition, algorithms and systems - Selected papers from GREC'97*, volume 1389, pages 390–410. Lecture Notes in Computer Science (LNCS), 1998.

[136] A. Cornuejols, L. Miclet, Y. Kodratoff, and T. Mitchell. *Apprentissage artificiel : Concepts et algorithmes*. Eyrolles, 2002. Collection : Technique scientifique télécom.

[137] S. Derrode, M. Daoudi, and F. Ghorbel. Invariant content-based image retrieval using a complete set of Fourier-Mellin descriptors. *Int. Conf. on Multimedia Computing and Systems (ICMCS'99)*, pages 877–881, 1999.

[138] J. Dougherty, R. Kohavi, and M. Sahami. *Supervised and unsupervised discretization of continuous features*. Morgan Kaufman, 1995.

[139] A. Frank and A. Asuncion. UCI machine learning repository, 2010. http://archive.ics.uci.edu/ml.

[140] G. Gasmi, S. Ben-Yahia, E. Mephu-Nguifo, and Y. Slimani. IGB : une nouvelle base générique informative des règles d'association. *I3, Sciences du Traitement de l'Information*, 2006.

[141] T. Hamrouni, S. Ben Yahia, and Y. Slimani. Avoiding the itemset closure computation "pitfall". In *Proc. of the 3rd International Conference on Concept Lattices and their Applications (CLA'05)*, pages 46–59, 2005.

[142] J. Han and K. Kamber. *Data-Mining : concepts and techniques*. Morgan Kaufmann, 2001.

[143] D. Hand, H. Mannila, and P. Smyth. *Principles of data-mining*. The MIT Press, 2001.

[144] M. Houtsma and A. Swami. Set-oriented mining for association rules in relational databases. In IEEE Computer Society Press, editor, *Proc. of 11th International Conference on Data Engineering (ICDE'95)*, pages 25–33, March 1995.

[145] M.-K. Hu. Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*, 8(2) :179–187, Feb. 1962.

[146] T. Kanungo, R.M. Haralick, H.S. Baird, W. Stuetzle, and D. Madigan. Document degradation models : parameter estimation and model validation. In *IAPR Workshop on machine vision applications (MV'94)*, pages 552–557, 1994.

[147] G. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2) :119–127, 1980.

[148] M. Klimushkin, S. Obiedkov, and C. Roth. Approaches to the selection of relevant concepts in the case of noisy data. In *Proc. of 8th International Conference on Formal Concept Analysis (ICFCA'10)*, 2010.

[149] D.J. Krus and E.A. Fuller. Computer assisted multicrossvalidation in regression analysis. *Educational and Psychological Measurement*, 42 :187–193, 1982.

[150] M. Kryszkiewicz. *Concise representations of frequent patterns and association rules*. PhD thesis, Warsaw Univeristy of Technologie, 2002.

[151] S. Kuznetsov. Machine learning on the basis of formal concept analysis. *Automation and Remote Control archive*, 62(10) :1543–1564, 2001.

[152] S. Kuznetsov. Machine learning and formal concept analysis. In *Proc. of International conference on industrial and engineering applications of artificial intelligence and expert systems*, volume 3029, pages 287–312, Ottawa, Canada, 2004. Lecture notes in computer science : Innovations in applied artificial intelligence.

[153] S. Kuznetsov, S. Obiedkov, and C. Roth. Reducing the representation complexity of lattice-based taxonomies. In *Proc. of International Conference on Computational Science (ICCS'07)*, 2007.

[154] W. Li, J. Han, and J. Pei. CMAR : Accurate and efficient classification based on multiple class-association rules. In *Proc. of IEEE International Conference on Data Mining (ICDM'01)*, pages 369–376, San Jose, CA, 2001. IEEE Computer Society.

[155] M. Liquière and E. Mephu-Nguifo. LEGAL : LEarning with GAlois Lattice. In *Actes des Journées Françaises sur l'Apprentissage (JFA'90)*, pages 93–113, Lannion, France, avril 1990.

[156] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of the fourth International Conference on Knowledge Discovery and Data Mining (KDDM'98)*, pages 80–86, New York, USA, 1998.

[157] J. Lladós, E. Valveny, and G. Sanchez. A case study of pattern recognition : Symbol recognition in graphic documents. In Jean-Marc Ogier and Eric Trupin, editors, *Pattern Recognition in Information Systems*, pages 1–13. Proc. of the 3rd International Worshop on Pattern Recognition in Information Systems, ICEIS Press, 2003.

[158] J. Lladós, E. Valveny, G. Sanchez, and E. Marti. Symbol recognition : Current advances and perspectives. In *Graphics recognition, algorithms and applications - Selected papers from GREC'01*, volume 2390, pages 104–127. Lecture Notes in Computer Science (LNCS), 2002.

[159] H. Mannila, H. Toivonen, and A.I. Verkamo. Efficient algorithms for discovering association rules. In AAI Press, editor, *Proc. of Workshop on Knowledge Discovery in Databases (KDD'94)*, pages 181–192, July 1994.

[160] E. Mephu-Nguifo. Une nouvelle approche basée sur le treillis de Galois, pour l'apprentissage de concepts. *Mathématiques et Sciences Humaines*, 124 :19–38, 1993.

[161] E. Mephu-Nguifo and P. Njiwoua. Treillis des concepts et classification supervisée. *Technique et Science Informatiques, RSTI*, 24(4) :449–488, 2005. Hermès - Lavoisier, Paris, France.

[162] T. Mitchell. *Machine learning*. McGraw-Hill, 1997.

[163] P. Njiwoua and E. Mephu-Nguifo. Améliorer l'apprentissage à partir d'instances grâce à l'induction de concepts : le système CIBLe. *Revue d'intelligence Artificielle (RIA)*, 13(2) :413–440, 1999. Hermès Science.

[164] G. Oosthuizen. Supergran : A connectionist approach to learning, integrating genetic algorithms and graph induction. In *Proc. of International Conference on Genetic Algorithms (GEM'87)*, pages 132–139, 1987.

[165] G. Oosthuizen. *The use of a Lattice in Knowledge Processing*. PhD thesis, University of Strathclyde, Glasgow, 1988.

[166] G. Oosthuizen. Machine learning : a mathematical framework for neural network, symbolic and genetics-based learning. In *Proc. of the third international conference on Genetic algorithms (GEM'89)*, pages 385–390, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[167] G. Oosthuizen. Rough sets and concept lattices. In *Proc. of the International Workshop on Rough Sets and Knowledge Discovery (RSKD'93)*, pages 24–31, London, UK, 1994. Springer-Verlag.

[168] N. Pasquier. *Data mining : Algorithmes d'extraction et de réduction des règles d'association dans les bases de données*. PhD thesis, Université of Clermont-Ferrand II, 2000.

[169] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information System*, 24(1) :25–46, 1999.

[170] J. R. Quinlan. *C4.5 : programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[171] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1, 1986.

[172] J. Radon. Uber die Bestimmung von Funktionen durch ihre Integralwerte langs gewisser Mannigfaltigkeiten. *Berichte Saechsische Akademie der Wissenschaften*, pages 262–277, 1917.

[173] R. Rakotomalala. *Graphes d'induction*. PhD thesis, Université C. Bernard, Lyon I, Dec. 1997.

[174] R. Rakotomalala. Arbres de décision. *Revue MODULAD*, 33 :163–187, 2005.

[175] Graphic Recognition. Base d'images grec 2003 (graphics recognition), 2003. http://www.cvc.uab.es/grec2003/SymRecContest/index.htm, date visite : 08-01-2008.

[176] Graphic Recognition. Base d'images grec 2005 (graphics recognition), 2005. http://www.cs.cityu.edu.hk/grec2005, date visite : 08-01-2008.

[177] M.A. Rodrigues. *Invariants for pattern recognition and classification*, volume 42 of *Machine Perception Artificial Intelligence*. World Scientific, 2000.

[178] M. Sahami. Learning classification rules using lattices. In Nada Lavrac and Stefan Wrobel, editors, *Proc. of the International Conference on Machine Learning (ICML'95)*, pages 343–346, Heraclion, Crete, Greece, April 1995.

[179] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithms for mining association rules in large databases. In Morgan Kaufmann, editor, *Proc. of the 21st International Conference on very Large Databases (VLDB'95)*, pages 432–444, Sept. 1995.

[180] C. Silverstein, S. Brin, and R. Motwani. Betond market baskets : generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2(1) :39–68, 1998.

[181] G. Stumme, R. Taouil, Y. Bastide, and L. Lakhal. Computing iceberg concept lattices with titanic. In *Proc. of Data and Knowledge Engineering (DKE'02)*, pages 189–222, 2002.

[182] S. Tabbone and L. Wendling. Adaptation de la transformée de Radon pour la recherche d'objets à niveaux de gris et de couleurs. *Technique et Science Informatique*, 22(9) :1041–1068, 2003.

[183] S. Tabbone, L. Wendling, and K. Tombre. Matching of graphical symbols in line-drawing images using angular signature information. *International Journal on Document Analysis and Recognition*, 6(2) :115–125, Oct. 2003.

[184] M. Teague. Image analysis via the general theory of moments. *Journal of Optical Society of America (JOSA)*, 70 :920–930, 1980.

[185] K. Tombre and B. Lamiroy. Graphics recognition - from re-engineering to retrieval. *Proc. of 7th ICDAR, Edinburgh (Scotland, UK)*, pages 148–155, August 2003.

[186] N. Topsze. *Treillis de Galois et réseaux de neurones : une approche constructive d'architecture des réseaux de neurones*. PhD thesis, Université d'Artois, Nov. 2010.

[187] O.D. Trier, A.K. Jain, and T. Taxt. Features extraction methods for character recognition - a survey. *Pattern Recognition*, 29 :641–662, 1996.

[188] T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databaes. In *Proc. of the 7th International Conference on Discovery Science (DS'04)*, 2004.

[189] E. Valveny and P. Dosch. Symbol recognition contest : A synthesis. In J. Lladós and Y.B. Kwon, editors, *Graphics Recognition : Recent Advances and Perspectives - Selected papers from GREC'03*, volume 3088, pages 368–385. Lecture Notes in Computer Science, 2004.

[190] J. Wang and G. Karypis. HARMONY : Efficiently mining the best rules for classification. In *Proc. of the International Conference of Data Mining (ICDM'05)*, Newport Beach, CA, 2005.

[191] J.H. Xiaoxin Yin. CPAR : Classification based on predictive association rules. In *Proc. of the International Conference on Data Mining (ICDM'03)*, pages 369–376, San Francisco, CA, 2003.

[192] Z. Xie, W. Hsu, Z. Liu, and M. Lee. Concept lattice based composite classifiers for high predictability. *Journal of Experimental and Theoretical Artificial Intelligence*, 14(2/3) :143–156, 2002.

[193] Z. Xie, W. Hsu, Z. Liu, and M. Lee. Concept lattice based composite classifiers for high predictability. *Journal of Experimental and Theoretical Artificial Intelligence*, 14(2/3) :143–156, 2002.

[194] D. Zuwala. *Reconnaissance de symboles sans connaissance a priori*. PhD thesis, Institut National Polytechnique de Lorraine, 6 novembre 2006.

Relations spatiales et ontologies

[195] Napoli A. and Le Ber F. The Galois lattice as a hierarchical structure for topological relations. *Annals of Mathematics and Artificial Intelligence*, 49 :171–190, 2007.

[196] S. Adam. *Interprétation de Documents Techniques : des Outils à leur Intégration dans un Système à Base de Connaissances*. PhD thesis, Université de Rouen, 11 décembre 2001.

[197] F. Baader. Computing a minimal representation of the subsumption lattices of all conjunctions oc concepts defined in a terminology. In *First International Sympsosium on Knowledge Retrieval, Use and Storage for Efficiency (KRUSE'95)*, 1995.

[198] F. Baader and R. Molitor. Building and restructuring description logic knowledge bases using least common subsumers and concept analysis. In *Proc. of the 8th International Cconference on Conceptual Structures (ICCS'00)*, pages 292–305, 2000.

[199] F. Baader and W. Nutt. *The Description Logic Handbbook : Theory, Implementation and Applications*, chapter Basic Description Logics. Cambridge University Press, 2003.

[200] F. Baader and U. Sattler. Tableau algorithms for description logics. *Studia Logica*, 69, 2001.

[201] F. Baader and B. Sertjkaya. Applying formal concept analysis to description logics. In *Proc. of the 2nd International Conference on Formal Concept Analysis (ICFCA'04)*, pages 261–286, 2004.

[202] D. T. Barnard, L. Burnard, J.-P. Gaspart, L. A. Price, C. M. Sperberg-McQueen, and G. B. Varile. Hierarchical encoding of text : Technical problems and sgml solutions. *Computers and the Humanities*, 29(3) :211–231, 1995. http://www.tei-c.org/index.xml.

[203] F. Le Ber and L. Mangelinck. A formal representation of landscape spatial patterns to analyze satellite images. *Engineering Applications of Artificial Intelligence*, 12(1-3) :51–59, 1998.

[204] F. Le Ber and A. Napoli. Design and comparison of lattices of topological relations based on galois lattice theory. In *Proc. of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR'02)*, pages 37–48. Morgan Kaufmann, Apr. 22-25 2002.

[205] F. Le Ber and A. Napoli. Design and comparison of lattices of topological relations for spatial representation and reasoning. *Exp. Theor. Artif. Intell.*, 15(3) :331–371, 2003.

[206] T. Berners-Lee, J. Hendlers, and O. lassila. The semantic web. *Scientific American*, 284(5) :34–43, 2001.

[207] R.J. Brachman and J.G. Schmolze. An overview of the kl-one knowledge representations system. *Cognitive Sciences*, 9(2) :171–216, 1985.

[208] H. Bunke. Graph-based representations in document analysis. In *Tenth International Conference on Document Analysis and Recognition (ICDAR'09)*, 2009.

[209] B.-L. Clarke. A calculus of individuals based on connection. *Notre dame journal of formal Logic*, 22(3) :204–218, 1981.

[210] A.G. Cohn, B. Bennet, J. Gooday, and N.M. Gotts. Representing and reasoning with qualitative spatial relations about regions. In *Spatial and temporal reasoning*, pages 97–134. Kluwer, 1997.

[211] L.P. Cordella and M. Vento. Symbol recognition in documents : a collection of techniques ? *International Journal on Document Analysis and Recognition (IJDAR)*, 3(2) :73–88, 2000.

[212] M. Coustaty, J.-M. Ogier, R. Pareti, and N. Vincent. Drop Caps Decomposition For Indexing - A New Letter Extraction Method. In *Proc. of the IAPR 10th International Conference on Document Analysis and Recognition (ICDAR'09)*, pages 476–480, Barcelona Espagne, 2009.

[213] M. Delalandre. *Analyse de documents graphiques : une approche par reconstruction d'objets*. PhD thesis, Université de Rouen, Rouen, France, 2005.

[214] M. Delalandre, P. Heroux, S. Adam, E. Trupin, and J.M. Ogier. A statistical and structural approach for symbol recognition using XML modelling. *Structural and Syntactical Pattern Recognition (SSPR)*, 2396 :281–290, 2002. LNCS Springer.

[215] M. Delalandre, E. Trupin, J.M. Ogier, and J. Labiche. Contextual system of symbol structural recognition based on an object-process methodology. *Electronic Letters on Computer Vision and Image Analysis*, 5(2) :16–29, 2005.

[216] A. Deruyver, Y. Hodé, and J.M. Jolion. Pyramide adaptative et graphe sémantique : Un processus de segmentation dirigé par la connaissance. In *15ème édition du congrès Reconnaissance des Formes et Intelligence Artificielle (RFIA'06)*, 2006.

[217] F. M. Domini. *The description logic handbook : theory, implementation and applications*, chapter Complexity of reasoning, pages 96–136. Cambridge University Press, 2003.

[218] P. Dosch and J. Lladós. Vectorial signatures for symbol discrimination. In J. Llados and Y.B. Kwon, editors, *Graphics Recognition : Recent Advances and Perspectives - Selected papers from GREC'03*, volume 3088, pages 154–165. LNCS, 2004.

[219] S. Dubois, M. Lugiez, R. Péteri, and M. Ménard. Adding a noise component to a color decomposition model for improving color texture extraction. In *proc. of the 4th European Conference on Colour in Graphics, Imaging, and Vision (CGIV'08)*, pages 394–398, 2008.

[220] M. Eigenhofer. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2) :161–174, 1991.

[221] A. Etemadi, J.-P. Schmidt, G. Matas, J. Illingworth, and J. Kittler. Low-Level Grouping of Straight Line Segments. *British Machine Vision Conference*, pages 119–126, 1991.

[222] H. Freeman. On the encoding of arbitrary geometric configurations. *Transactions on Electronic and Computer*, 10 :260–268, 1961.

[223] B. Ganter. Attribute exploration with background knowledge. *Theoretical Computer Science*, 217 :215–233, 1999.

[224] P. Geibel and F. Wysotzki. Learning relationnal concepts with decision trees. In Lorenza Saitta, editor, *12th European Conference on Artificial Intelligence (ECAI'96)*, pages 166–174, San Fransisco, 1996. Morgan Kaufmann Publishers.

[225] M. Grigni, D. Papadias, and C. Padadimitriou. Topological inference. In *International Joint Conferences on Artificial Intelligence (IJCAI'95)*, pages 901–906, 1995.

[226] T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2) :199–220, 1993.

[227] V. Haarslev and R. Möller. RACER system description. In *Proc. of the International Joint Conference on Automated Reasoning (IJCAR'01)*, 2001.

[228] A. El Hamidi, M. Ménard, M. Lugiez, and C. Ghannam. Weighted and extended total variation for image restoration and decomposition. *Pattern Recognition*, 43(4) :1564 – 1576, 2010.

[229] I. Horrocks. Using an expressive description logic : FaCT or fiction ? In *Proc. of the 6th International Conference on the Principles of Knowledge Representation and reasoning (KR'98)*, pages 636–647, 1998.

[230] I. Horrocks, P.F. Patel-Schneider, and F.v. Harmelen. From SHIQ and RDF to OWL : The making of a web ontology language. *Journal of Web Semantics*, 1(1) :7–26, 2003.

[231] I. Horrocks, P.F. Patel-Schneider, and F.v. Harmelen. An introduction to description logics. *The description logic handbook : theory, implementation, and applications*, pages 1–40, 2003.

[232] P.V.C. Hough. Machine Analysis of Bubble Chamber Pictures. In *Proc. of International Conference on High Energy Accelerators and Instrumentation*, pages 554–556, 1959.

[233] P.V.C. Hough. Method and means for recognizing complex patterns. U.S. Patent 3069654, Dec. 1962.

[234] Q. Iqbal and J.K. Aggarwal. Retrieval by Classification of Images Containing Large Manmade Objects Using Perceptual Grouping. *Pattern recognition*, 35 :1463–1479, 2002.

[235] Jena. Jena - a semantic web framework for java, 2011. http://jena.sourceforge.net/, date visite : Mars 2011.

[236] R. Jimenes. Les bibliothèques virtuelles humanistes et l'étude du matériel typographique. Technical report, Centre d'Etude Superieur de la Renaissance, 2008.

[237] H. Knublauch, R.W. Fergerson, N.F. Noy, and M.A. Musen. The protégé owl plugin : An open development environment for semantic web applications. In

International Semantic Web Conference (ISWC), volume 3298, pages 229–243, 2004.

[238] J. Llados, E. Marti, and J. J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *PAMI*, 23(10) :1137 – 1143, 2001.

[239] J. Llados and G. Sanchez. Symbol recognition using graphs. In *Proc. of International Conference on Image Processing (ICIP'03)*, pages 49–52, 2003.

[240] H. Locteau, S. Adam, E. Trupin, J. Labiche, and P. Héroux. Symbol recognition combining vectorial and statistical features. In Liu Wenyin and Josep Lladós, editors, *Graphics Recognition : Ten years Review and Future Perspectives - Selected papers from GREC'05*, volume 3926, pages 76–87. Lecture Notes in Computer Science, 2005.

[241] S. W. Lu, Y. Ren, and Ch. Suen. Hierarchical attributed graph representation and recognition of handwritten chinese characters. *Pattern Recognition*, 24(7) :617–632, 1991.

[242] M. Luqman, M. Delalandre, T. Brouard, J.Y. Ramel, and J. Llados. Employing fuzzy intervals and loop-based methodology for designing structural signature : an application to symbol recognition. In *Proc. of Graphics Recognition (GREC'09)*, pages 22–31. Lecture Notes in Computer Science, 2009.

[243] Eigenhofer M. A formal definition of binary topological relationships. In Berlin Springer, editor, *Proc. of Foundations of Data Organization and Algorithms (FODO'89)*, volume Lncs 367, pages 457–472, 1989.

[244] L. Mangelinck. *Représentation et classification de structures spatiales. Application à la reconnaissance de paysages agricoles*. PhD thesis, Univeristé Nancy 1, 1998.

[245] B. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30 :45–87, 1981.

[246] B.T. Messmer and H. Bunke. Clustering and error-corrrecting matching of graphs for learning and recognition of symbols in engineering drawings. *Document Analysis Systems II*, pages 102–117, 1998.

[247] B. Nebel. Reasoning and revision in hybrid representation systems. In Springer-Verlag, editor, *Lecture Notes In Artificial Intelligence*, 1990.

[248] R. Pareti and N. Vincent. Ancient initial letters indexing. In *Proc. of the 18th International Conference on Pattern Recognition (ICPR'08)*, pages 756–759, Hong Kong, China, 2006. IEEE Computer Society.

[249] P.F. Patel-Schneider, P. Hayes, and I. Horrocks. Owl web ontology language semantics and abstract syntax, 2004. http://www.w3.org/TR/2004/, visité le 10/02/2004.

[250] W.K. Pratt. *Digital Image Processing : PIKS Scientific Inside*. Wiley-Interscience, 4 edition, February 2007.

[251] S. Prediger and G. Stumme. Theory driven logical scaling : conceptual information systems meet description logics. In *Proc. of the sixth International Worksho on Knowledge Representation meets Databases (KRDB'99)*, 1999.

[252] R.J. Qureshi, J.Y. Ramel, and H. Cardot. Graphic symbol recognition using flexible matching of attributed relationnal graphs. In *Proc. of the 6th IASTED International Conference on Visualization, Imaging and Image Processing (VIIP'06)*, pages 383–388, 2006.

[253] D.A. Randell, Z. Cui, and A.G. Cohn. A spatial logic based on regions and connection. In Morgan Kaufmann, editor, *Proc. of the 3rd International Conference on Knowledge representation and Reasoning (KR'92)*, 1992.

[254] R. Raveaux, E. Barbu, H. Locteau, S. Adam, P. Héroux, and E. Trupin. A graph classification approach using a multi-objective genetic algorithm application to symbol recognition. *Proc. of IAPR International Workshop in Graph-Based Representations in Pattern Recognition (GBRPR'07)*, 4538 :361–370, 2007.

[255] J. Rocha and T. Pavlidis. Character recognition without segmentation. *PAMI*, 17(9) :903–909, 1995.

[256] M.H. Rouane, M. Huchard, A. Napoli, and P. Valtchev. A proposal for combining formal concept analysis and description logics for mining relational data. In *Proc. of the fifth International Conference on Formal Concept Analysis (ICFCA'07)*, pages 51–65, Clermont-Ferrand, France, Feb. 2007.

[257] M. Rusiñol, A. Borràs, and J. Lladós. Relational indexing of vectorial primitives for symbol spotting in line-drawing images. *Pattern Recognition Letters*, 2009.

[258] G. Sánchez and J. Lladós. Syntactic models to represent perceptually regular repetitive patterns in graphic documents. In *Graphic Recognition (GREC'03)*, pages 166–175, 2003.

[259] U. Sattler. Description logic reasoners, 2011. http://www.cs.man.ac.uk/~sattler/reasoners.html, dernière visite : Avril 2011.

[260] K. Schild. A correspondance theory for terminological logics : preliminary report. In *Proc. of the 12th International Conference on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.

[261] B. Sertkaya. *Formal Concept Analysis Methods for Description Logics*. PhD thesis, Technischen Universität Dresden, Jul. 2008.

[262] B. Sertkaya. A survey on how description logic ontologies benefit from formal concept analysis. In *Proceedings of Fifth International Conference on Concept Lattices and their Applications (CLA'07)*, Séville, Espagne, Oct. 2010.

[263] J. Sharma, D.-M. Flewelling, and M.-J. Egenhofer. A qualitative spatial reasoner. In *Proc. of the 6th International Sympsosium on Spatial Data Handling (SDH'94)*, pages 665–681, 1994.

[264] E. Sirin and B. Parsia. Pellet : an OWL DL reasoner. In *Proc. of the International Workshop on Description Logic (DL'04)*, 2004.

[265] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner : System description. In *Proc. of the International Joint Conference on Automated Reasoning (IJCAR'06)*, pages 292–297, 2006.

[266] M. Uschold and M. King. Ontologies : principles, methods and applications. *Knowledge Engineering Review*, 11(2) :93–155, 1996.

[267] H.V.d. Waal. *ICONCLASS, an Iconographic Classification System*. Completed and edited by L. D. Couprie et al, 1985. http://www.iconclass.nl/.

[268] L. Wenyin, W. Zhang, and L. Yan. An interactive example-driven approach to graphics recognition in engineering drawings. *International Journal on Document Analysis and Recognition (IJDAR)*, 2006.

[269] L. Yan and L. Wenyin. Interactive recognition of graphic objects in engineering drawings. In J. Lladós and Y.B. Kwon, editors, *Graphics Recognition : Recent Advances and Perspectives - Selected papers from GREC'03*, volume 3088, pages 128–141. Lecture Notes in Computer Science, 2003.

[270] D. Zuwala and S. Tabbone. Une méthode de localisation et de reconnaissance de symboles sans connaissance a priori. In Laurence Likforman-Sulem, editor, *Neuvième Colloque International Francophone sur l'Ecrit et le Document (CIFED'06)*, pages 127–131, Fribourg, Suisse, Sept. 2006.