Fouille de Base de Données

Version 2.0

Alain Casali
alain.casali@univ-amu.fr

@AMU - faites vous plaisir!.

Introduction

La fouille de données [1] dans les bases de données désigne le processus non trivial d'extraction d'informations implicitement enfouies dans les bases de données. Les informations extraites sont des règles, des modèles, des concepts, des régularités, des anomalies ou des tendances qui sont utiles, intéressantes et intelligibles du point de vue de l'utilisateur. Le data mining doit son succès à l'essor des techniques de collecte et de stockage des données. Ces dernières ont conduit à la création de bases de données de taille importante (plusieurs giga-octets [2]) qui contiennent implicitement une grande quantité d'informations intéressantes. L'extraction de ces informations présente un intérêt majeur pour le secteur industriel ou scientifique. L'analyse, par exemple, des bases de données de transactions des ventes d'un supermarché permettra d'étudier le comportement des clients et de réorganiser les rayons afin d'améliorer les ventes. En analysant les données d'organismes de vente par correspondance, il est possible de regrouper des clients selon certains critères, ce qui permettra

1. également connu sous le nom de data mining ou KDD (Knowledge Discovery in Databases

2. source : http://www.wintercorp.com/vldb/2003_TopTen_Survey/TopTenWinners.asp

ensuite de limiter les coûts de « *mailing* » en définissant de manière plus précise les clients potentiels. L'analyse des bases de données médicales ou génétiques permet de mettre en évidence des corrélations, ce qui peut préciser de nouveaux axes de recherche.

En résumé, on peut voir le data mining comme une aide à la décision. Le processus essaye de mettre en évidence des informations cachées dans la base de données. Ces informations peuvent être très diverses suivant ce qui intéresse l'utilisateur.

Le data mining est un processus itératif et semi-automatique constitué de plusieurs étapes allant de la sélection à la préparation des données jusqu'à l'interprétation des résultats. Les différentes étapes de ce processus sont représentées sur la figure 1.

Après un bref rappel mathématique sur les concepts nécessaires à la compréhension de ce manuel, nous porterons notre attention, dans le deuxième chapitre, sur le modèle relationnel en caractérisant les ensembles maximaux (i) dans un premier temps à partir d'un ensemble de dépendances fonctionnelles $\mathcal{F}$, puis (ii) à partir d'une relation base de données r. Les maximaux jouent un rôle important pour la fouille de données. En effet, une fois ceux-ci calculés, nous pouvons aisément déduire une couverture canonique $\mathcal{F}'$ de l'ensemble de dépendances fonctionnelles $\mathcal{F}$, calculer les clés minimales de la relation, obtenir une relation r' de taille réduite et vérifiant uniquement les dépendances fonctionnelles valides sur r. De plus, les maximaux nous permettent de savoir, en un temps polynomial, si la relation r est en 3NF ou en BCNF.

Dans, le troisième chapitre, nous étudierons les principes généraux des algorithmes par niveaux et appliquerons ceux-ci pour l'extraction des minimaux transversaux et des motifs fréquents. L'approche par niveaux se justifie aisément car les bases de données sont de plus en plus volumineuses et ne tiennent plus dans la mémoire centrale de l'ordinateur. Le but d'un algorithme par niveaux est de minimiser le nombre d'accès à la base de données tout en offrant un temps de réponse satisfaisant. Nous verrons l'algorithme *Apriori* qui est l'algorithme de référence pour l'extraction des motifs fréquents, l'algorithme *Pascal*, qui est une optimisation « *presque optimale* » d'*Apriori* se basant sur le concept de classes d'équivalences et l'algorithme *Close* qui se base sur une

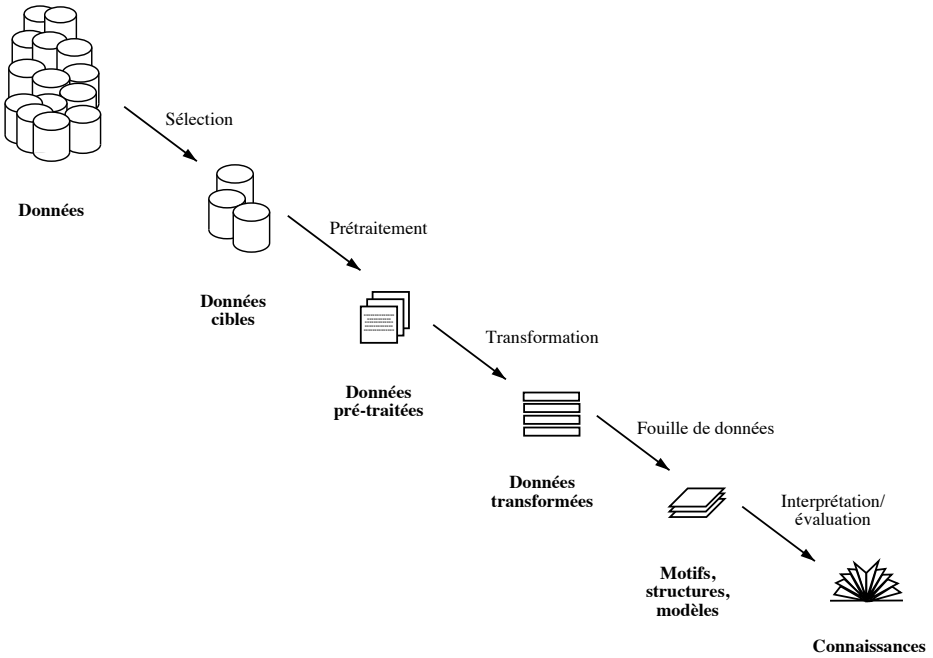


FIGURE 1: Étapes du processus de fouille de données

approche clés-fermés. Nous découvrirons comment nous pouvons générer toutes les règles d'association valides à partir des motifs fréquents et extraire un sous-ensemble des règles nous permettant de retrouver les autres.

Pour finir, nous découvrirons, dans le troisième chapitre, deux méthodes de classification : (*i*) l'espace de versions et (*ii*) la classification bayesienne. L'espace de versions est la seule technique offrant une classification avec une précision de 100%. Quant à la classification bayesienne, c'est la plus simple à mettre en œuvre et celle qui offre le meilleur rapport « *Simplicité / Efficacité* ».

Sommaire

I Fouille de base de données binaires **1**

1 Règles d'association **3**

1.1 Introduction aux algorithmes par niveaux 4

1.2 Génération des motifs fréquents 7

1.3 Extraction des règles d'association 18

1.4 Close : une alternative pour l'extraction des motifs fréquents . 23

1.5 Extraction de règles d'association informatives 28

2 Classification **33**

2.1 Espace de Versions 33

2.2 Classification Bayesienne Simple 37

II Fouille de base de données n-aires **41**

3 Dépendances Fonctionnelles **43**
3.1 Rappels sur les dépendances fonctionnelles 44
3.2 Ensembles maximaux et dépendances fonctionnelles 52
3.3 Inférence des dépendances fonctionnelles 58
3.4 Relation d'Armstrong 61
3.5 Clés minimales d'une relation 63
3.6 Tests de formes normales 64

4 Conclusion **69**

III Annexes **73**

5 Annexe **75**
5.1 Les ensembles ordonnés 75
5.2 Notions de Treillis 84
5.3 Calcul des fermés d'un treillis 92

Bibliographie **97**

Index **101**

I

Fouille de base de données binaires

Chapitre

1

Sommaire

1.1 Introduction aux algorithmes par niveaux
1.2 Génération des motifs fréquents
1.3 Extraction des règles d'association
1.4 Close : une alternative pour l'extraction des motifs fréquents
1.5 Extraction de règles d'association informatives

Règles d'association

DANS CE CHAPITRE, nous nous proposons de résoudre le problème suivant : « *Étant donné une relation binaire r sur un ensemble d'items (attributs) $\mathcal{I}$ et deux seuils minsup et minconf donnés en paramètre, trouvez tous les sous-ensembles X de $\mathcal{I}$ tels que la fréquence de X dans r soit supérieure ou égale à minusport et tels que la confiance de la règle $X \Rightarrow Y \backslash X$ soit supérieure ou égale à minconf (Y est aussi un motif fréquent et un sûr-ensemble de X)* ». Dans cette optique, nous introduirons un nouveau type d'algorithme : les algorithmes par niveaux. Leur principe est de balayer l'espace de recherche (le treillis des parties de $\mathcal{I}$) tout en minimisant le nombre d'accès à la base de données. En effet, accéder à un élément qui réside en mémoire centrale « *coûte* » environ 10^{-7} seconde alors qu'accéder à un élément stocké sur disque « *coûte* » environ 10^{-2} seconde. Encore une fois, cette approche se justifie car les bases de données les plus volumineuses atteignent presque les 30 Go de données [1]. Nous étudierons l'algorithme *Apriori* [1] qui est l'algorithme de référence pour ce problème. Puis nous verrons l'algorithme *Pascal* [3] qui est une optimisation « *presque optimale* » d'*Apriori* ainsi que l'algorithme *Close* [25] qui permet de calculer des motifs fermés fréquents. Comme nous le constaterons sur des

1. source : http://www.wintercorp.com/vldb/2003_TopTen_Survey/TopTenWinners.asp

exemples, le nombre de règles d'association peut être énorme, c'est pourquoi nous chercherons à réduire ce nombre en calculant la base informative réduite [3] qui est une couverture des règles d'association.

1.1 Introduction aux algorithmes par niveaux

Qu'est ce qu'un algorithme par niveaux?

Presque tous les problèmes de Data Mining sont des problèmes NP-Complets et nécessitent l'exploration d'un treillis des parties (dépendant du problème). Dans notre cas, c'est le treillis des parties de $\mathcal{I}$ qui doit être exploré. Nous allons parcourir ce treillis en ne considérant, en même temps, que les sous-ensembles de $\mathcal{I}$ ayant même cardinalité. On examine tous les sous-ensembles de cardinalité 1, puis ceux de cardinalité 2, ..., puis $|\mathcal{I}|$.

1.1.1 Génération des candidats pour un algorithme par niveaux

La génération du niveau supérieur est une étape prépondérante pour les algorithmes par niveaux puisque chaque élément n'est généré qu'une et une seule fois pour gagner du temps d'exécution. Cette génération est basée sur l'opérateur semi-union, noté $\sqcup$, que l'on peut définir ainsi :

Définition 1.1 Opérateur semi-union
Soit $X = A_1A_2...A_n$ et $Y = B_1B_2...B_n \subseteq \mathcal{I}$ deux éléments de même cardinalité. Alors :

$$Z = X \sqcup Y \Leftrightarrow Z = \begin{cases} X \cup Y \text{ ssi } A_1A_2...A_{n-1} = B_1B_2...B_{n-1} \text{ et } A_n <_{lex} B_n \\ \emptyset \text{ sinon.} \end{cases}$$

Exemple 1.1 Si $X = ABC$ et $Y = ABD$, alors $X \sqcup Y = ABCD$ et $Y \sqcup X = \emptyset$. $A \sqcup B = AB$ car seule la deuxième condition portant sur l'opérateur semi-union est alors vérifiée.

1.1.2 Algorithme par niveaux et contrainte monotone

Pour une contrainte monotone (*cmc*), dès que nous aurons trouvé un motif satisfaisant cette contrainte, alors tous ses sur-ensembles satisferont aussi la contrainte *cmc*. Donc, pour chaque niveau i, nous allons avoir un ensemble de motifs candidats C_i ainsi qu'un ensemble de motifs qui vérifient la contrainte L_i. Les éléments du niveau suivant (C_{i+1}) seront générés à partir de ceux de $C_i \backslash L_i$. Ce sont les éléments de C_i qui ne satisfont pas la contrainte *cmc*. On itère ce processus jusqu'à ce qu'il n'y ait plus aucun élément dans C_i. De plus, il ne faudra pas oublier de vérifier qu'il n'existe aucun sous-ensemble de l'élément généré dans L_i.

Nous allons appliquer ce principe pour la découverte des minimaux transversaux d'un hypergraphe (donc d'un ensemble E ou d'une relation binaire r) car la contrainte « $X \subseteq \mathcal{I}$ *est un minimal transversal de* r *(ou de* $\overline{r}$*)* » est une contrainte monotone. L'algorithme TR (niveaux) nous permet de trouver les minimaux transversaux de r ou de $\overline{r}$.

Alg. 1 TR (niveaux)

Entrée : $\mathcal{I}$ un ensemble d'attributs, r une relation binaire
Sortie : $Tr(r)$ ou $Tr(\overline{r})$
1: $C_1 := \{A \subseteq \mathcal{I} : |A| = 1\}$
2: $i := 1$
3: **tant que** $C_i \neq \emptyset$ **faire**
4: $\quad L_i := \{X \in C_i \mid X \cap t \neq \emptyset,\ \forall t \in r\}$ // *$X \not\subseteq t$ si on cherche $Tr(\overline{r})$*
5: $\quad C_{i+1} := \{Z = X \sqcup Y \mid X, Y \in C_i \backslash L_i, Z \neq \emptyset \text{ et } \forall W \lessdot Z, W \in C_i \backslash L_i\}$
6: $\quad i := i + 1$
7: **fin tant que**
8: **return** $\bigcup_i L_i$

Exemple 1.2 Soit $\mathcal{I} = ABCDE$ et $r = \{D, AB, BC, BE\}$. Regardons la trace de l'algorithme TR dans le cas d'une recherche des minimaux transversaux.

- Niveau 1 :
  $C_1 = \{A, B, C, D, E\}$
  $L_1 = \{\emptyset\}$

- Niveau 2 :
  $C_2 = \{AB, AC, AD, AE, BC, BD, BE, CD, CE, DE\}$
  $L_2 = \{BD\}$

- Niveau 3 :
  $C_3 = \{ABC, ABE, ACD, ACE, BCE, CDE\}$
  Bien que généré dans un premier temps, $ABD \notin C_3$ car $BD \subset ABD$ et $BD \in L_2$. Il en sera de même pour tous les sur-ensembles de BD pour ce niveau.
  $L_3 = \{\emptyset\}$

- Niveau 4 :
  $C_4 = \{ABCE, ACDE\}$
  $L_4 = \{ACDE\}$

- Niveau 5 :
  $C_5 = \{\emptyset\}$

Par conséquent, $Tr(r) = \{BD, ACDE\}$.

Exercice 1.1
Vérifier que l'on obtient bien le même résultat que dans l'exemple 3.16 du chapitre précédent.

1.1.3 Algorithme par niveaux et contrainte antimonotone

Pour une contrainte monotone (*camc*), si un motif $X \subseteq \mathcal{I}$ satisfait la contrainte, alors tous ses sous-ensembles vérifient aussi la contrainte. Donc, pour chaque niveau i, nous allons avoir un ensemble de motifs candidats C_i ainsi qu'un ensemble de motifs qui vérifient la contrainte L_i. Les éléments du niveau suivant (C_{i+1}) seront générés à partir de ceux de L_i, tout en vérifiant que tous les sous-ensembles des éléments générés sont bien dans L_i. On itère ce processus jusqu'à ce qu'il n'y ait plus aucun élément dans C_i.

Un modèle « *générique* » d'algorithme par niveaux pour les contraintes antimonotones est le suivant :

Alg. 2 Algorithme par niveaux généralisé pour les contraintes antimonotones

Entrée : $\mathcal{I}$ un ensemble d'attributs, r une relation binaire, *camc* une contrainte antimonotone

Sortie : ensemble des motifs $X \subseteq \mathcal{I}$ vérifiant la contrainte

1: $C_1 := \{A \subseteq \mathcal{I} : |A| = 1\}$
2: $i := 1$
3: **tant que** $C_i \neq \emptyset$ **faire**
4: $L_i := \{X \in C_i \mid camc(X)\}$\\ balayer la relation r pour vérifier la condition
5: $C_{i+1} := \{Z = X \sqcup Y \mid \forall X, Y \in L_i, Z \neq \emptyset$ et $\forall W \lessdot Z, W \in L_i\}$
6: $i := i + 1$
7: **fin tant que**
8: **return** $\bigcup_i L_i$

1.2 Génération des motifs fréquents

Dans cette partie, nous allons nous consacrer à la résolution de la première partie de notre problème : « *Soit r une relation binaire sur un ensemble d'items (attributs) $\mathcal{I}$ et un seuil minsup donné en paramètre, trouvez tous les sous-ensembles X de $\mathcal{I}$ tels que la fréquence d'apparition de X dans r soit supérieure ou égale à minusp* ». Nous allons voir l'approche *Apriori*, puis une optimisation avec l'algorithme *Pascal*.

1.2.1 Approche *Apriori*

Cette approche est une application directe de l'algorithme 2. Mais, avant d'étudier cet algorithme, définissons plus formellement la fréquence d'un motif.

Définition 1.2 Fonction fréquence
Soit $\mathcal{I}$ un ensemble d'items sur une relation binaire r. On définit une fonction f comme suit :

$$f : \mathcal{P}(\mathcal{I}) \to \mathcal{P}(RowId)$$
$$X \mapsto \{t \in RowId \mid X \subseteq t[\mathcal{I}]\}$$

On définit la fréquence d'un motif $X \subseteq \mathcal{I}$ comme étant le rapport entre la cardinalité de $f(X)$ et le nombre de tuples de la relation binaire, soit $Freq(X) =$

$\frac{|f(X)|}{|r|}$. Étant donné un seuil $minsup \in [0, 1]$ donné par l'utilisateur, si $Freq(X) \geq minsup$ alors le motif X est fréquent. Notons par $\mathbb{F}$ l'ensemble des motifs fréquents.

Propriété 1.1 La fonction $Freq$ définie précédemment est une fonction monotone décroissante, *i.e.* si $X \subset Y$, alors $Freq(X) \geq Freq(Y)$.

Exemple 1.3 Soit la relation binaire r_1 suivante :

RowId	Item
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

TABLE 1.1: Relation exemple r_1

Nous avons $f(A) = \{1, 3, 6\}$ et donc $Freq(A) = 3/6$. De même $f(AB) = \{3, 5\}$ et $Freq(AB) = 2/6$.

Exercice 1.2
En considérant la relation r_1, calculez $f(BC)$, $f(D)$ et $f(ABCDE)$.

Puisque la contrainte « $X \subseteq \mathcal{I}$ *est un motif fréquent sur* r » est une contrainte antimonotone, nous pouvons utiliser l'algorithme 2 en remplaçant la contrainte. Nous obtenons l'algorithme *Apriori*.

Alg. 3 Algorithme *Apriori*

Entrée : $\mathcal{I}$ un ensemble d'attributs, r une relation binaire et un seuil $minsup$
Sortie : $\mathbb{F}$ ensemble des motifs fréquents

1: $C_1 := \{A \subseteq \mathcal{I} : |A| = 1\}$
2: $i := 1$
3: **tant que** $C_i \neq \emptyset$ **faire**
4: $\quad L_i := \{X \in C_i \mid Freq(X) \geq minsup\}$\\ balayer la relation r pour vérifier la condition
5: $\quad C_{i+1} := \{Z = X \sqcup Y \mid \forall X, Y \in L_i, Z \neq \emptyset$ et $\forall W \lessdot Z, W \in L_i\}$
6: $\quad i := i + 1$
7: **fin tant que**
8: **return** $\bigcup_i L_i$

Exemple 1.4 Avec la relation r_1 et en considérant $minsup = 2/6$, nous obtenons l'exécution suivante de l'algorithme *Apriori* :

- Niveau 1 :
  $C_1 = \{A, B, C, D, E\}$

L_1	$Freq$
A	3/6
B	5/6
C	5/6
E	5/6

  Puisque D n'est pas fréquent, aucun de ses sur-ensembles ne sera considéré à l'avenir.

- Niveau 2 :
  $C_2 = \{AB, AC, AE, BC, BD, CD, CE\}$

L_2	$Freq$
AB	2/6
AC	3/6
AE	2/6
BC	4/6
BE	5/6
CE	4/6

- Niveau 3 :
  $C_3 = \{ABC, ABE, ACE, BCD\}$

L_3	$Freq$
ABC	2/6
ABE	2/6
ACE	2/6
BCE	4/6

- Niveau 4 :
  $C_4 = \{ABCE\}$

L_4	$Freq$
ABCE	2/6

Ainsi, $\mathbb{F} = \{A, B, C, E, AB, AC, AE, BC, BE, CE, ABC, ABE, ACE, BCE, ABCE\}$.

De plus, si $|\max_{\subseteq}(\mathbb{F})| > 1$ alors l'ensemble ordonné $\langle \mathbb{F} \cup \mathcal{I}, \subseteq \rangle$ est un treillis appelé **treillis des motifs fréquents**. Remarquons que nous avons dû ajouter $\mathcal{I}$ à l'ensemble des motifs fréquents pour « *fermer la structure* ». Par contre si $|\max_{\subseteq}(Freq(r))| = 1$, alors l'ensemble ordonné $\langle \mathbb{F}, \subseteq \rangle$ est un treillis aussi appelé treillis des items fréquents. Ce treillis comporte des opérateurs $\bigwedge$ et $\bigvee$ qui sont respectivement l'intersection ($\cap$) et la fermeture de l'union ($h(\cup)$). Nous pouvons considérer l'opérateur de fermeture défini page 90 dans la définition 5.15. Ceci sera exposé plus en détail dans la partie suivante.

Exemple 1.5 Avec la relation r_1 proposée dans l'exercice 1.3, voici le treillis des motifs fréquents que nous obtenons :

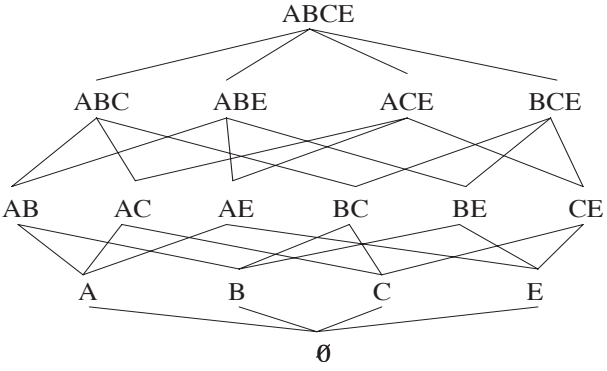


FIGURE 1.1: Treillis des motifs fréquents de r_1.

Exercice 1.3
Soit r_2 la relation suivante. Calculer l'ensemble de motifs fréquents pour $minsup = 2/6$ en utilisant l'algorithme *Apriori*.

RowId	Item
1	BCDE
2	ABC
3	ABF
4	CDE
5	ACD
6	AEF

TABLE 1.2: Relation r_2

1.2.2 Approche *Pascal*

Il est possible de minimiser le nombre de balayages de la base de données en utilisant le concept de classes d'équivalence : on regroupe au sein de la même

classe d'équivalence tous les motifs présents dans les mêmes tuples. De plus, ce concept nous offre une méthode originale pour la dérivation de la fréquence d'un motif.

Définition 1.3 Classe d'équivalence
Soit r une relation binaire sur $\mathcal{I}$ et $X \subseteq \mathcal{I}$. On définit une classe d'équivalence, notée [.], comme suit : $[X] = \{Y \subseteq \mathcal{I} \mid f(X) = f(Y)\}$, f étant la fonction spécifiée dans la définition 1.2. $[X]$ est la classe d'équivalence de X.

Exemple 1.6 Avec la relation r_1 proposée dans l'exemple 1.3, nous avons $[A] = [AC] = \{A, AC\}$ et $[C] = \{C\}$.

Ainsi, si deux motifs $X, Y \subseteq \mathcal{I}$ appartiennent à la même classe d'équivalence, alors ils auront la même fréquence dans r; i.e. $[X] = [Y] \Rightarrow Freq(X) = Freq(Y)$. De plus, si X est inclus dans Y et si X et Y ont même fréquence, alors ils appartiennent à la même classe d'équivalence; i.e. $X \subseteq Y$ et $Freq(X) = Freq(Y) \Rightarrow [X] = [Y]$.

Remarque : Une erreur souvent commise est de penser que si deux motifs ont même fréquence alors ils appartiennent à la même classe d'équivalence; i.e. $Freq(X) = Freq(Y) \not\Rightarrow [X] = [Y]$. C'est grâce à la condition d'inclusion que l'implication est vérifiée. Si on veut l'omettre, il faudrait utiliser une implication du type $Freq(X) = Freq(Y) = Freq(X \cup Y) \Rightarrow [X] = [Y]$, mais il reste à la démontrer ...

Exemple 1.7 Nous pouvons dessiner les classes d'équivalence sur la figure précédente (cf. Figure 1.1) et obtenons celle-ci :

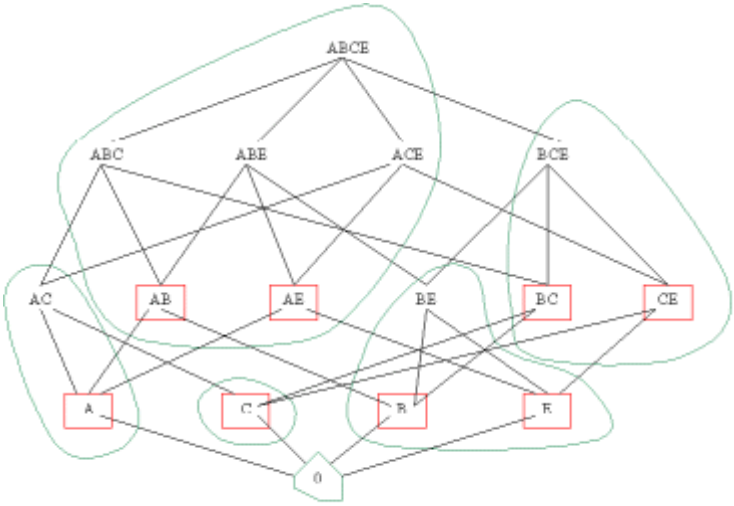


FIGURE 1.2: Treillis des classes d'équivalences des motifs fréquents de r_1.

Les clés sont encadrées et les classes d'équivalences sont entourées dans ce treillis.

Puisque les classes d'équivalence ne sont pas connues à l'avance, nous allons les construire en utilisant une approche par niveaux. Nous allons déterminer les motifs minimaux (selon l'inclusion) des classes d'equivalence car la contrainte « $X \subseteq \mathcal{I}$ *est un minimal d'une classe d'équivalence* » est une contrainte antimonotone; *i.e.*$\forall\ Y \subset X$, Y est aussi un minimal d'une classe d'équivalence (mais pas celle de X).

Classe d'équivalence et système de fermeture

D'après la définition d'une clé, un motif X est clé si et seulement si $X \in \min_{\subseteq}(\{Y \subseteq \mathcal{I} \mid h(X) = h(Y)\})$, h étant l'opérateur de fermeture défini sur une relation binaire page 90. Or cet opérateur de fermeture est fortement lié à la fonction f, définie dans la partie 1.2.1, car cette fonction est une composante de la connexion de Galois [4, 11] servant à définir h d'une autre manière que celle

exposée dans le chapitre consacré aux rappels. Nous allons étendre la définition de clé à celle de minimaux des classes d'équivalence au travers de la proposition suivante :

Proposition 1.1 Soit $X \subseteq \mathcal{I}$ un ensemble d'items, r une relation binaire et h l'opérateur « *classique* » de fermeture sur r. Nous disons que X est une clé de $h(X)$ si et seulement si $X \in \min_{\subseteq}([h(X)]) \Leftrightarrow X \in \min_{\subseteq}([X])$.

Corollaire 1.1 Sous les conditions de la proposition précédente, nous avons :

$$\max_{\subseteq}([X]) = h(X).$$

Exemple 1.8 Avec la relation r_1 proposée dans l'exemple 1.3, nous avons $[A] = [AC] = \{A, AC\}$. Donc $h(A) = AC$ et $Key(AC) = \{A\}$.

L'algorithme Pascal

Le problème restant à résoudre est l'inférence de la fréquence des motifs qui sont fréquents mais qui ne sont pas clés. Pour ce faire, nous allons utiliser les classes d'équivalence via la propriété suivante :

Proposition 1.2 Soit $X \subseteq \mathcal{I}$ un motif fréquent et r une relation binaire sur $\mathcal{I}$, si X est un motif non clé, alors $Freq(X) = \min(\{Freq(X \backslash x) \mid x \in X\})$

Remarquons que dans certains cas, il nous faudra appliquer récursivement cette proposition.

Exemple 1.9 Avec la relation r_1 proposée dans l'exemple 1.3, nous savons que AC n'est pas clé puisque $h(A) = AC$, nous nous demandons alors quelle est sa fréquence? Puisque la fréquence de C est 5/6 et celle de A est 3/6, la proposition précédente implique que la fréquence de AC est 3/6 puisque c'est la plus petite des deux.

Exercice 1.4
Dérivez la fréquence du motif fréquent $ABCE$ à partir des clés fréquentes.

Ainsi, nous allons scinder l'ensemble C_i de l'algorithme *Apriori* en deux sousensembles distincts : (i) un ensemble de motifs non clés dont nous pouvons dériver la fréquence de chaque élément en utilisant la proposition 1.2 et que

nous « *injecterons* » directement dans L_i et (ii) un ensemble C_i qui correspond aux clés candidates du niveau i pour lesquelles il va falloir balayer la base de données afin de trouver leur fréquence.

Alg. 4 Algorithme *Pascal*

Entrée : $\mathcal{I}$ un ensemble d'attributs, r une relation binaire et un seuil $minsup$
Sortie : ensemble des motifs fréquents

1: $C_1 = \{A \subseteq \mathcal{I} : |A| = 1\}$
2: $i = 1$
3: **tant que** $C_i \neq \emptyset$ **faire**
4: balayer la base de données pour trouver la fréquence des éléments de C_i
5: $L_i^* = \{X \in C_i \mid Freq(X) \geq minsup\}$
6: **pour tout** $X \in L_i^*$ **faire**
7: **si** $\forall x \in X, Freq(X \backslash x) \neq Freq(X)$ **alors**
8: $Key(X) = vrai$
9: **sinon**
10: $Key(X) = faux$
11: **fin si**
12: **fin pour**
13: $L_i := L_i \cup L_i^*$
14: $C_{i+1} := \{Z = X \sqcup Y \mid \forall X, Y \in L_i, Z \neq \emptyset \text{ et } \forall W \lessdot Z, W \in L_i\}$
15: **pour tout** $X \in C_{i+1}$ **faire**
16: **si** $\exists Y \in L_i \mid Y \subset X$ et $Key(Y) = faux$ **alors**
17: $Key(X) = faux$
18: $Freq(X) = \min(\{Freq(Y) \mid \forall Y \in L_i \text{ et } Y \subset X\})$
19: $L_{i+1} := L_{i+1} \cup X$
20: $C_{i+1} := C_{i+1} \backslash X$
21: **fin si**
22: **fin pour**
23: $i := i + 1$
24: **fin tant que**
25: **return** $\bigcup_i L_i$

Exemple 1.10 Avec la relation r_1 et en considérant $minsup = 2/6$, nous obtenons l'exécution suivante de l'algorithme *Pascal* :

- Niveau 1 :
  $C_1 = \{A, B, C, D, E\}$

L_1^*	$Freq$	Key
A	3/6	vrai
B	5/6	vrai
C	5/6	vrai
E	5/6	vrai

- Niveau 2 :
  $C_2 = \{AB, AC, AE, BC, BD, CD, CE\}$

L_2^*	$Freq$	key
AB	2/6	vrai
AC	3/6	faux
AE	2/6	vrai
BC	4/6	vrai
BE	5/6	faux
CE	4/6	vrai

- Niveau 3 :
  $C_3 = \{ABC, ABE, ACE, BCD\}$

L_3	$Freq$
ABC	2/6
ABE	2/6
ACE	2/6

Le fait que AC et CE ne soient pas des clés nous permet de dériver le support de ces items de niveau 3.

L_3^*	$Freq$	Key
BCE	4/6	faux

- Niveau 4 :
  $C_4 = \{ABCE\}$

L_4	$Freq$
ABCE	2/6

Comme avec l'exécution d'*Apriori*, nous obtenons $\mathbb{F} = \{A, B, C, E, AB, AC, AE, BC, BE, CE, ABC, ABE, ACE, BCE, ABCE\}$.

Exercice 1.5
Utiliser l'algorithme *Pascal* sur la relation r_2 (cf. tableau ??) avec le seuil $minsup = 2/6$.

1.2.3 Résultats expérimentaux

Soit les jeux de données suivants :

Nom	Nombre de tuples	Taille moyenne motifs par tuple	Nombre d'items
C20D10K	10 000	20	386
C73D10K	10 000	73	2 178
Mushrooms	8 416	23	128

TABLE 1.3: Descriptif des bases expérimentales

Comparons les temps d'exécution des algorithmes *Pascal* et *Apriori* afin de vérifier que *Pascal* est bien une optimisation utile d'*Apriori* dans la pratique :

Fréquence	Nombre de fréquents	*Pascal*	*Apriori*
20,0 %	20 239	9,44	57,15
15,0 %	36 359	12,31	85,35
10,0 %	89 883	19,29	164,81
7,5 %	153 163	23,53	232,40
5,0 %	352 611	33,06	395,32
2,5 %	1,160 363	55,33	754,64

TABLE 1.4: Temps de réponse pour C20D10K

Fréquence	Nombre de fréquents	*Pascal*	*Apriori*
80 %	109 159	177,49	3 661,27
75 %	235 271	392,80	7 653,58
70 %	572 087	786,49	17 465,10
60 %	4 355 543	3 972,10	109 204,00

TABLE 1.5: Temps de réponse pour C73D10K

Fréquence	Nombre de fréquents	*Pascal*	*Apriori*
20,0 %	53 337	6,48	115,82
15,0 %	99 079	9,81	190,94
10,0 %	600 817	23,12	724,35
7,5 %	936 247	32,08	1 023,24
5,0 %	4 140 453	97,12	2 763,42

TABLE 1.6: Temps de réponse pour MUSHROOMS

1.3 Extraction des règles d'association

Cette partie du chapitre est consacrée exclusivement à la résolution du problème suivant : « *Étant donné un ensemble de motifs fréquents* $\mathbb{F}$ *sur une relation* r *et un seuil* $minconf \in [0,1]$, *trouver toutes les règles d'association du type* $X \Rightarrow Y \backslash X$ ($X \subset Y \subseteq \mathcal{I}$ *et* $Y \in \mathbb{F}$) *telles que la confiance de la règle* $X \Rightarrow Y \backslash X$ *soit supérieure ou égale à* $minconf$ ». La confiance d'une règle est définie comme suit :

Définition 1.4 Confiance d'une règle
Soit $Y \subseteq \mathcal{I}$, alors la confiance de la règle $X \Rightarrow Y \backslash X$ notée $conf(X \Rightarrow Y \backslash X)$, est le rapport entre la fréquence de Y et celle de X, soit :

$$conf(X \Rightarrow Y \backslash X) = \frac{Freq(Y)}{Freq(X)}.$$

Étant donné un seuil $minconf$, la règle $X \Rightarrow Y \backslash X$ est valide si et seulement si $conf(X \Rightarrow Y \backslash X) \geq minconf$. Cette règle est exacte si et seulement si $conf(X \Rightarrow Y \backslash X) = 1$, sinon elle est approximative.

La fréquence d'une règle correspond à la force de cette règle et la confiance à sa précision. D'une manière probabiliste, on peut dire que $Freq(X \Rightarrow Y \backslash X)$ correspond à $Proba[X \cup Y]$ et $conf(X \Rightarrow Y \backslash X)$ correspond à $Proba[Y \mid X]$.

Nous allons exposer un algorithme d'extraction de toutes les règles d'association qui est assez simple. Des algorithmes plus optimisés sont donnés dans [25]. L'idée de base est de construire, pour chaque motif fréquent $X \in \mathbb{F}$, l'ensemble $\mathbb{F}'$ qui est composé de tous les sur-ensembles Y de X ainsi que leur fréquence. Ensuite, pour chaque élément de $\mathbb{F}'$, nous déduisons la confiance de la règle $X \Rightarrow Y \backslash X$ en appliquant la formule précédente. Nous itérons ce processus jusqu'à ce que tous les motifs fréquents aient été pris en compte.

Alg. 5 Algorithme d'extraction des règles d'association

Entrée : $\mathcal{I}$ un ensemble d'attributs, $\mathbb{F}$ ensemble des motifs fréquents sur la relation binaire r et un seuil $minconf$
Sortie : ensemble des règles d'association

1: **pour tout** $X \in \mathbb{F}$ **faire**
2: $\quad \mathbb{F}' := \{Y \in \mathbb{F} \mid X \subset Y\}$
3: $\quad$ **pour tout** $Y \in \mathbb{F}'$ **faire**
4: $\quad\quad$ calculer $conf(X \Rightarrow Y \backslash X) = \frac{Freq(Y)}{Freq(X)}$
5: $\quad\quad$ **si** $conf(X \Rightarrow Y \backslash X) \geq minconf$ **alors** marquer la règle comme valide
6: $\quad$ **fin pour**
7: **fin pour**

Exemple 1.11 Avec la relation r_1 et en considérant $minconf = 2/3$, nous obtenons l'exécution suivante de l'algorithme précédent :

- $\underline{X = A :}$ $\mathbb{F}' = \{AB, AC, AE, ABC, ABE, ACE, ABCE\}$

Règle	confiance	validité ?
$A \Rightarrow B$	2/3	oui
$A \Rightarrow C$	1	oui
$A \Rightarrow E$	2/3	oui
$A \Rightarrow BC$	2/3	oui
$A \Rightarrow BE$	2/3	oui
$A \Rightarrow CE$	2/3	oui
$A \Rightarrow BCE$	2/3	oui

- $\underline{X = B :}$ $\mathbb{F}' = \{B, AB, BC, BE, ABC, ABE, BCE, ABCE\}$

Règle	confiance	validité ?
$B \Rightarrow A$	2/5	non
$B \Rightarrow C$	4/5	oui
$B \Rightarrow E$	1	oui
$B \Rightarrow AC$	3/5	non
$B \Rightarrow AE$	2/5	non
$B \Rightarrow CE$	4/5	oui
$B \Rightarrow ACE$	2/5	non

- $\underline{X = C :}$ $\mathbb{F}' = \{C, AC, BC, CE, ABC, ACE, BCE, ABCE\}$

Règle	confiance	validité ?
$C \Rightarrow A$	3/5	non
$C \Rightarrow B$	4/5	oui
$C \Rightarrow E$	4/5	oui
$C \Rightarrow AB$	2/5	non
$C \Rightarrow AE$	2/5	non
$C \Rightarrow BE$	4/5	oui
$C \Rightarrow ABE$	2/5	non

- $\underline{X = E :}$ $\mathbb{F}' = \{E, AE, BE, CE, ABE, ACE, BCE, ABCE\}$

Règle	confiance	validité ?
$E \Rightarrow A$	$2/5$	non
$E \Rightarrow B$	1	oui
$E \Rightarrow C$	$4/5$	oui
$E \Rightarrow AB$	$2/5$	non
$E \Rightarrow AC$	$2/5$	non
$E \Rightarrow BC$	$4/5$	oui
$E \Rightarrow ABC$	$2/5$	non

- $\underline{X = AB :}$ $\mathbb{F}' = \{ABC, ABE, ABCE\}$

Règle	confiance	validité ?
$AB \Rightarrow C$	1	oui
$AB \Rightarrow E$	1	oui
$AB \Rightarrow CE$	1	oui

- $\underline{X = AC :}$ $\mathbb{F}' = \{ABC, ACE, ABCE\}$

Règle	confiance	validité ?
$AC \Rightarrow B$	$2/3$	oui
$AC \Rightarrow E$	$2/3$	oui
$AC \Rightarrow BE$	$2/3$	oui

- $\underline{X = AE :}$ $\mathbb{F}' = \{ABE, ACE, ABCE\}$

Règle	confiance	validité ?
$AE \Rightarrow B$	1	oui
$AE \Rightarrow C$	1	oui
$AE \Rightarrow BC$	1	oui

- $\underline{X = BC :}$ $\mathbb{F}' = \{ABC, BCE, ABCE\}$

Règle	confiance	validité ?
$BC \Rightarrow A$	1/2	non
$BC \Rightarrow E$	1	oui
$BC \Rightarrow AE$	1/2	non

- $\underline{X = BE :}$ $\mathbb{F}' = \{ABE, BCE, ABCE\}$

Règle	confiance	validité ?
$BE \Rightarrow A$	2/5	non
$BE \Rightarrow C$	4/5	oui
$BE \Rightarrow AC$	1/3	non

- $\underline{X = CE :}$ $\mathbb{F}' = \{ACE, BCE, ABCE\}$

Règle	confiance	validité ?
$CE \Rightarrow A$	1/2	non
$CE \Rightarrow B$	1	oui
$CE \Rightarrow AB$	1/2	non

- $\underline{X = ABC :}$ $\mathbb{F}' = \{ABCE\}$

Règle	confiance	validité ?
$ABC \Rightarrow E$	1	oui

- $\underline{X = ABE :}$ $\mathbb{F}' = \{ABCE\}$

Règle	confiance	validité ?
$ABE \Rightarrow C$	1	oui

- $\underline{X = ACE :}$ $\mathbb{F}' = \{ABCE\}$

Règle	confiance	validité ?
$ACE \Rightarrow B$	1	oui

- $\underline{X = BCE :}$ $\mathbb{F}' = \{ABCE\}$

Règle	confiance	validité ?
$BCE \Rightarrow A$	1/2	non

Nous avons donc généré 14 règles exactes et 17 règles approximatives.

Exercice 1.6
Utiliser l'algorithme d'extraction des règles d'association sur la relation r_2 (cf. tableau **??**) avec le seuil $minsup = 2/6$ et la confiance $minconf = 1/3$ pour trouver toutes les règles d'association valides.

Comme nous pouvons l'entr'apercevoir au travers de l'exemple précédent, le nombre de règles d'association peut être énorme. Si nous considérons la base de données C73D10K, $minsup = 90\%$ et $minconf = 80\%$, alors plus de 2 000 000 de règles sont générées. Ce nombre étant trop élevé pour pouvoir être exploité, nous verrons, dans le paragraphe 1.5, comment nous pouvons calculer une couverture des règles d'association.

1.4 Close : une alternative pour l'extraction des motifs fréquents

Vu que le nombre de motifs fréquents peut être gigantesque (cf. tableau 1.5 et 1.6), il est important de pouvoir en extraire un sous-ensemble nous permettant de retrouver facilement la fréquence d'un motif fréquent. Dans un premier temps, nous allons étendre la notion de couverture, vue dans le chapitre **??**, à notre problème.

Définition 1.5 Couverture pour les motifs fréquents

Nous disons qu'un ensemble $\mathbb{G}$ est une couverture pour $\mathbb{F}$ si et seulement si :

1. $\mathbb{G} \subseteq \mathbb{F}$
2. $\mathbb{G} \vDash Freq(X), \forall X \in \mathbb{F}$

Autrement dit, nous devons être capables de dériver la fréquence de tout motif fréquent en ne connaissant que la fréquence des éléments de $\mathbb{G}$.

En utilisant le corollaire 1.1 qui fait le lien entre les maximaux des classes d'équivalence et le fermé ($\max_{\subseteq}([X]) = h(X)$), nous pouvons affirmer que l'ensemble des fermés fréquents est une couverture pour $\mathbb{F}$.

Theorème 1.1
L'ensemble des fermés fréquents, noté $CL(\mathbb{F})$, est une couverture pour $\mathbb{F}$.

Le corollaire suivant nous permet de (i) savoir si un motif est fréquent et (ii) si tel est le cas, de retrouver sa fréquence.

Corollaire 1.2 Soit $CL(\mathbb{F})$ l'ensemble des fermés fréquents et $X \subseteq \mathcal{I}$ un motif, alors :

1. $X \in \mathbb{F} \Leftrightarrow \exists\, Y \in \max_{\subseteq}(CL(\mathbb{F})) \mid X \subseteq Y$.
2. si X est fréquent, alors $Freq(X) = Freq(Y) \mid Y = \min_{\subseteq}(CL(\mathbb{F}))$ et $X \subseteq Y$.

Exemple 1.12 Avec la relation r_1 et en considérant $minsup = 2/6$, nous avons vu que nous avons $\mathbb{F} = \{A, B, C, E, AB, AC, AE, BC, BE, CE, ABC, ABE, ACE, BCE, ABCE\}$. De plus $CL(\mathbb{F}) = \{C, AC, BE, BCE, ABCE\}$. Dans notre cas, $\max_{\subseteq}(CL(\mathbb{F})) = ABCE$. Donc, le motif D n'est pas un motif fréquent car $D \nsubseteq ABCE$. Par contre A est un motif fréquent car $A \subseteq ABCE$. Calculons sa fréquence. Il nous faut connaître le plus petit fermé fréquent contenant A (ceci correspond à la partie « $Y = \min_{\subseteq}(CL(\mathbb{F}))$ et $X \subseteq Y$ » du corollaire précédent). Dans notre cas, $Y = AC$. Puisque la fréquence de AC est $1/2$, alors celle de A est aussi $1/2$.

Cependant, pour calculer $CL(\mathbb{F})$, nous n'allons pas utiliser la méthode de Norris car il nous faudrait stocker le treillis des fermés en mémoire et donc disposer d'une mémoire « *infinie* » [2]. De même, nous n'appliquerons pas l'algorithme Next-Closure car l'ordre lectique n'est pas compatible avec la contrainte

2. Cette supposition n'est jamais vraie dans la vie réelle.

de fréquence. Sur la relation r_1, D n'est pas fréquent, il est donc inutile de calculer sa fermeture ainsi que celle de chacun de ses sur-ensembles. Cependant, puisque E est fréquent, il nous faut calculer la fermeture des motifs contenant E et nous savons que les fermés contenant E seront générés après ceux contenant D. De plus, nous devrons aussi calculer la fermeture de DE tout en sachant que ce motif n'est pas fréquent. Á méditer ... Nous allons modifier l'algorithme *Pascal* en ne conservant que la partie sur la génération des clés et de leurs supports respectifs. Une fois toutes les clés fréquentes calculées, nous balayons une et une seule fois la base de données pour calculer la fermeture de chaque clé. Le fermé ainsi obtenu aura pour fréquence celle de sa clé (cf. corollaire 1.1).

Alg. 6 Algorithme *Close*

Entrée : $\mathcal{I}$ un ensemble d'attributs, r une relation binaire et un seuil $minsup$
Sortie : ensemble des fermés fréquents

1: $C_1 := \{A \subseteq \mathcal{I} : |A| = 1\}$
2: $i := 1$
3: **tant que** $C_i \neq \emptyset$ **faire**
4: balayer la base de données pour trouver la fréquence des éléments de C_i
5: $L_i := \{X \in C_i \mid Freq(X) \geq minsup\}$
6: **pour tout** $X \in L_i$ **faire**
7: **si** $\exists x \in X \mid Freq(X \backslash x) = Freq(X)$ **alors** $L_i := L_i \backslash X$
8: **fin pour**
9: $C_{i+1} := \{Z = X \sqcup Y \mid \forall X, Y \in L_i, Z \neq \emptyset$ et $\forall W \lessdot Z, W \in L_i\}$
10: $i := i + 1$
11: **fin tant que**
12: $Key := \bigcup_i L_i \cup \{\emptyset\}$
13: **pour tout** $X \in Key$ **faire**
14: **return** $h(X)$ et $Freq(X)$
15: **fin pour**

Exemple 1.13 Avec la relation r_1 et en considérant $minsup = 2/6$, nous obtenons l'exécution suivante de l'algorithme *Close* :

- Niveau 1 :
  $C_1 = \{A, B, C, D, E\}$

L_1	$Freq$	Key
A	3/6	vrai
B	5/6	vrai
C	5/6	vrai
E	5/6	vrai

- Niveau 2 :
  $C_2 = \{AB, AC, AE, BC, BD, CD, CE\}$

L_2	$Freq$	key
AB	2/6	vrai
AE	2/6	vrai
BC	4/6	vrai
CE	4/6	vrai

AC et BE n'appartiennent pas à L_2 car ce ne sont pas des clés fréquentes. En effet $Freq(A) = Freq(AC) = 3/6$ et $Freq(B) = Freq(E) = Freq(BE) = 5/6$.

- Niveau 3 :
  $C_3 = \{\emptyset\}$

Donc $Key = \{\emptyset, A, B, C, E, AB, AE, BC, CE\}$. Calculons la fermeture de chacune des clés.

X	$h(X)$	$Freq(h(X))$
$\emptyset$	$\emptyset$	1
A	AC	3/6
B	BE	5/6
C	C	5/6
E	BE	5/6
AB	ABCE	2/6
AE	ABCE	2/6
BC	BCE	4/6
BE	BCE	4/6

TABLE 1.7: Ensemble des fermés fréquents de r_1 pour $minsup = 2/6$

Exercice 1.7
Utiliser l'algorithme *Close* sur la relation r_2 (cf. tableau ??) avec le seuil $minsup = 2/6$ pour extraire tous les fermés fréquents, leurs clés ainsi que leur fréquence.

1.4.1 Résultats expérimentaux

En considérant les trois bases décrites dans le tableau 1.3, nous avons ajouté les temps de réponse de l'algorithme *Close* afin de pouvoir le comparer avec *Pascal* et *Apriori*. Nous constatons que *Close* est beaucoup plus rapide que *Apriori* sans pour autant atteindre les temps d'exécutions de *Pascal*. Cependant, *Close* reste le seul algorithme efficace capable de calculer la base informative.

Support	Fréquents	*Pascal*	*Apriori*	*Close*
20,0 %	20 239	9,44	57,15	14,36
15,0 %	36 359	12,31	85,35	18,99
10,0 %	89 883	19,29	164,81	29,58
7,5 %	153 163	23,53	232,40	36,02
5,0 %	352 611	33,06	395,32	50,46
2,5 %	1,160 363	55,33	754,64	78,63

TABLE 1.8: Temps de réponse pour C20D10K

Support	Fréquents	*Pascal*	*Apriori*	*Close*
80 %	109 159	177,49	3 661,27	241,91
75 %	235 271	392,80	7 653,58	549,27
70 %	572 087	786,49	17 465,10	1 112,42
60 %	4 355 543	3 972,10	109 204,00	5 604,91

TABLE 1.9: Temps de réponse pour C73D10K

Support	Fréquents	*Pascal*	*Apriori*	*Close*
20,0 %	53 337	6,48	115,82	9,63
15,0 %	99 079	9,81	190,94	14,57
10,0 %	600 817	23,12	724,35	29,83
7,5 %	936 247	32,08	1 023,24	41,05
5,0 %	4 140 453	97,12	2 763,42	98,81

TABLE 1.10: Temps de réponse pour Mushrooms

1.5 Extraction de règles d'association informatives

L'approche *Close* qui calcule les clés et les fermés fréquents est une approche parfaite pour calculer la base informative [3] qui est une couverture de toutes les règles d'association valides. Le but de cette couverture est d'obtenir des règles dont la partie gauche (ou prémisse) est minimale (selon l'inclusion) et dont la partie droite (ou conséquent) est maximale (toujours selon l'inclusion).

1.5.1 Base informative exacte

Pour les règles exactes, nous avons $Conf(X \Rightarrow Y \backslash X) = 1$. D'après la definition de la confiance d'une règle (cf. définition 1.4) nous pouvons déduire l'égalité suivante : $Freq(X) = Freq(Y)$. De plus, puisque $X \subseteq Y$, alors X et Y appartiennent à la même classe d'équivalence. La prémisse des règles informatives exactes sera donc composée d'une clé et le conséquent sera composé du fermé relatif à cette clé moins celle-ci. Autrement dit, nous allons générer des règles du type $X \Rightarrow h(X) \backslash X \mid X \in Key$ (Key étant l'ensemble des clés fréquentes obtenu lors de l'utilisation de l'algorithme *Close*).

Exemple 1.14 La base informative exacte de la relation r_1 pour le seuil $minsup = 2/6$ est la suivante : $\mathcal{B} = \{A \Rightarrow C, B \Rightarrow E, E \Rightarrow B, AB \Rightarrow CE, AE \Rightarrow BC, BC \Rightarrow E, BE \Rightarrow C\}$. Nous avons donc 7 règles informatives exactes alors que nous avions 14 règles d'association exactes.

1.5.2 Base informative approximative

Avant de spécifier la base informative pour les règles approximatives, définissons les prédécesseurs d'un fermé fréquent X.

Définition 1.6 Opérateur DLB
Soit X un fermé fréquent ($X \in CL(\mathbb{F})$), alors l'ensemble des prédécesseurs directs de X (*Direct Lower Bound*) est défini comme suit : $DLB(X) = \max_{\subseteq}(\{Y \in CL(\mathbb{F}) \mid Y \subset X\})$. Donc $DLB(X)$ est constitué des motifs qui sont couverts par X dans le treillis des fermés fréquents.

Exemple 1.15 En considérant l'ensemble des fermés fréquents $CL(\mathbb{F})$ obtenu dans l'exemple 1.12, nous avons $DLB(ABCE) = \{AC, BCE\}$.

La base informative approximative est constituée des règles du type $Y \Rightarrow X \backslash Y \mid X \in CL(\mathbb{F})$ et $Y \in DLB(X)$. La confiance de cette règle s'obtient toujours en divisant la fréquence de Y par celle de X. Bien sûr, on ne conserve que les règles valides (*i.e.* celles dont la confiance est supérieure ou égale à $minconf$).

Exemple 1.16 La base informative approximative de la relation r_1 pour le seuil $minsup = 2/6$ et le seuil $minconf = 2/3$ est la suivante :

règle	confiance
$AC \Rightarrow BE$	$2/3$
$BE \Rightarrow C$	$4/5$
$C \Rightarrow BE$	$4/5$

TABLE 1.11: Base informative approximative de r_1

Nous n'avons plus que 3 règles informatives approximatives.

Exercice 1.8
Utiliser l'algorithme d'extraction des règles d'association informatives sur la relation r_2 (cf. tableau **??**) avec le seuil $minsup = 2/6$ et la confiance $minconf = 1/3$ pour trouver toutes les règles d'association informatives valides.

1.5.3 Dérivation de la confiance d'une règle à partir de la base informative

Il ne sert à rien de calculer la base informative si l'on est pas capable de retrouver la confiance d'une règle inconnue. La dérivation de la confiance d'une règle $X \Rightarrow Y \backslash X$ inconnue se fait en 3 étapes :

1. Calculer $h(X)$ et $h(Y)$.
2. Trouver la chaîne $S = \{h(X) = X_1, X_2, ..., X_n = h(Y)\} \mid X_i \in DLB(X_{i+1})$ $\forall i = 1..n$
3. Si $h(X) = h(Y)$ alors $conf(X \Rightarrow Y \backslash X) = 1$
   Sinon, $conf(X \Rightarrow Y \backslash X) = conf(h(X) \Rightarrow X_2 \backslash h(X)) * conf(X_2 \Rightarrow X_3 \backslash X_2) * ... * conf(X_i \Rightarrow X_{i+1} \backslash X_i) * ... * conf(X_{n_1} \Rightarrow h(Y) \backslash X_{n-1})$

Exemple 1.17 Abaissons la confiance pour cet exemple à 2/5. La base informative exacte ne change pas, la base informative approximative est la suivante :

règle	confiance
$AC \Rightarrow BE$	2/3
$BE \Rightarrow C$	4/5
$C \Rightarrow BE$	4/5
$C \Rightarrow A$	3/5
$BCE \Rightarrow A$	1/2

Calculons la confiance de la règle $C \Rightarrow AB$ à partir de la base informative approximative.

1. $h(C) = C$ et $h(ABC) = ABCE$
2. Nous pouvons construire 2 suites $S_1 = \{C, AC, ABCE\}$ et $S_2 = \{C, BCE, ABCE\}$.

3. Avec S_1 : $conf(C \Rightarrow AB) = conf(C \Rightarrow A) * conf(AC \Rightarrow BE) = 3/5 * 2/3 = 2/*5$ et avec S_2 nous obtenons $conf(C \Rightarrow AB) = conf(C \Rightarrow BE) * conf(BCE \Rightarrow A) = 4/5 * 1/2 = 2/*5$. Remarquons que nous retrouvons la confiance de la règle obtenue dans l'exemple 1.11.

1.5.4 Résultats expérimentaux

Le tableau 1.13 présente le nombre de règles d'association exactes et le nombre de règles informatives exactes pour les trois bases décrites dans le tableau 1.3. Seul le support varie car la confiance des règles est 1. Le gain obtenu avec la base informative varie d'un facteur allant de 5 à 50. Le tableau 1.13 présente le nombre de règles d'association extraites des trois bases. Cette fois-ci, nous faisons varier les seuils minimaux de confiance. Ce tableau illustre bien la réduction apportée par les règles informatives réduites, qui offre des gains d'un facteur de 40 à 500.

Jeu	*minsup*	Règles exactes	Règles exactes informatives
C20D10K	50 %	2 277	457
C73D10K	90 %	52 035	1 369
Mushrooms	30 %	7 476	543

TABLE 1.12: Règles d'association exactes et règles exactes informatives

Jeu (*minsup*)	*minconf*	Règles approximatives	Règles informatives réduites
C20D10K	70 %	89 601	1 957
(50 %)	30 %	116 791	1 957
C73D10K	90 %	2 053 896	5 718
(90 %)	80 %	2 053 936	5 718
Mushrooms	70 %	37 671	1 221
(30 %)	30 %	71 412	1 578

TABLE 1.13: Règles d'association approximatives et règles informatives réduites

Sommaire

2.1 Espace de Versions
2.2 Classification Bayesienne Simple

Chapitre

2

Classification

Contrairement au chapitre précédent, nous connaissons à l'avance le conséquent (ou la partie droite) des règles d'association que nous allons extraire. L'ensemble de ces règles sert à construire le classeur. Ensuite nous appliquons ces règles d'une certaine « *manière* », cette « *manière* » est définie dans le classifieur. Nous verrons deux modèles de classification : (i) l'espace de versions et (ii) la classification bayesienne.

2.1 Espace de Versions

Le cadre de l'espace de versions est basé sur un ordre partiel d'hypothèses dans un langage de concepts [22]. Un tel ordre partiel est dérivé de la généralité relative des hypothèses. Dans notre contexte, le langage de concepts est le cadre du trellis des parties des attributs et les hypothèses sont des tuples. Dans ce modèle, la classe $\mathcal{C}$ ne contient que deux valeurs notées '+' et '-'. Le classeur est construit en appliquant l'algorithme VSM (*Version Space Mining*) [7]. La particularité du classifieur est de pouvoir prédire avec exactitude la classe d'un nouveau tuple, s'il réussit à le classer.

2.1.1 Le classeur

Notons r^+ l'ensemble des tuples de la relation ayant la valeur '+' pour l'attribut $\mathcal{C}$ et r^- pour les autres (i.e. $r = r^+ \cup r^-$). Un motif X est dit consistant s'il est inclus dans tout les tuples de r^+ et n'est inclus dans aucun tuple de r^-, autrement dit X est consistant si et seulement si :

- $\forall t \in r^+, X \subseteq t$ (contrainte C_1)
- $\forall t \in r^-, X \nsubseteq t$ (contrainte C_2).

L'espace de versions, noté $VS(r)$, est composé de tous les tuples consistants. De plus, l'espace de versions est habituellement représenté par les bordures S et G qui contiennent respectivement le motif consistant maximal et les motifs consistants minimaux w.r.t. $\subseteq$ ($S = \max_{\subseteq}(VS(r))$ et $G = \min_{\subseteq}(VS(r))$). H. Hirsh [14] montre que tout ensemble satisfaisant la propriété de convexité est représentable par bordures. Or l'espace de versions d'une relation binaire est un espace convexe sur le treillis des parties des items et, par conséquent, préserve la représentation à l'aide des ensembles S et G.

La contrainte C_1 (resp. C_2) est équivalente à la contrainte $Freq(t, r^+) = 1$ (resp. $Freq(t, r^-) = 0$). La contrainte C_1 est antimonotone selon l'ordre d'inclusion (i.e. $X' \subseteq X$ et X satisfait $C_1 \Rightarrow t'$ satisfait C_1). La contrainte C_2 est monotone l'ordre d'inclusion (i.e. $X' \subseteq X$ et X' satisfait $C_2 \Rightarrow t$ satisfait C_2).

Puisque tout motif consistant doit satisfaire la contrainte C_1 ($Freq(X, r^+) = 1$), chaque motif de S doit être inclus dans la fermeture du motif $\emptyset$ sur r^+. De plus, les tuples consistants vérifient la contrainte C_2 ($Freq(t, r^-) = 0$), donc ces tuples doivent spécialiser les transversaux minimaux du complémentaire de $\overline{r^-}$. Or $VS(r)$ est un espace convexe ayant un seul motif maximal et chaque motif de $VS(r)$ a la même fréquence sur r ($\forall X \in VS(r), Freq(X, r) = |r^+|/|r|$), donc $VS(r)$ est une classe d'équivalence (cf. définition 1.3 p. 12). Ce résultat est énoncé dans le théorème 2.1.

Theorème 2.1
Soir r une relation binaire sur $\mathcal{I} \cup \mathcal{C}$ et $VS(r)$ son espace de versions. Alors nous pouvons caractériser les bordures S et G comme suit :

1. $G = Tr(\overline{r^-}, h(\emptyset, r^+))$

2. $S = h(g, r),\ g \in G$

Ce théorème est à la base de l'algorithme VSM :

Alg. 7 Algorithme VSM

Entrée : r^+, r^-
Sortie : S, G
1: calculer $S' := h(\emptyset, r^+)$
2: calculer $G := Tr(\overline{r^-})$
3: **pour tout** $g \in G$ **faire**
4: **si** $g \nsubseteq S'$ **alors** $G := G \backslash g$
5: **fin pour**
6: Calculer $S := h(g) \mid g \in G$

Exemple 2.1 Soit r la relation suivante :

RowId	Item	Classe
1	ACDEF	+
2	BCDEF	+
3	BCF	-
4	ACD	-

TABLE 2.1: relation exemple r

Examinons maintenant les différentes étapes de l'algorithme VSM.

1. $S' := ACDEF \cap BCDEF = CDEF$
2. $G := \{E, AB, AF, BD, DF\}$
3. $G := \{E, DF\}$ \\ on a éliminé AB car $AB \nsubseteq CDEF$ (idem pour AF et BD).
4. $S := h(E) := CDEF$

Il est impératif de calculer la dernière fermeture car il arrive que le motif S' calculé lors de la première étape soit un sur-ensemble de S.

2.1.2 Le classifieur

De plus, si la relation ne contient pas de données erronées [1] , les bordures S et G classent correctement un motif inconnu $X \subseteq \mathcal{I}$ comme suit :

$$Classe(X) = \begin{cases} '+' \text{ ssi } S \subseteq X. \\ '-' \text{ ssi } \forall\ g \in G, g \nsubseteq X. \\ \text{inconnu autrement.} \end{cases}$$

Exemple 2.2 Avec les bordures S et G précédemment obtenues, nous avons $Classe$
$(ABCDEF) =' +'$, $Classe(D) =' -'$ et $Classe(AE) = inconnu$.

Cette méthode de classification est peu utilisée dans la pratique car il n'est pas rare d'avoir des données erronées au sein de la même relation.

2.1.3 Requête SQL permettant de calculer les tuples consistants

Dans des SGBDR munis de fonctionnalités OLAP, tels que IBM DB2 ou Microsoft SQL Server, $VS(r)$ peut être calculé en utilisant l'opérateur Group By Cube [13] comme ceci :

SELECT A_1, ..., A_n
FROM r^+
GROUP BY CUBE (A_1, ..., A_n)
HAVING COUNT(*) $= |r^+|$
MINUS
SELECT A_1, ..., A_n
FROM r^-
GROUP BY CUBE (A_1, ..., A_n);

Exercice 2.1
Calculer les bordures S et G pour la relation suivante :

Á quelle classe appartiennent les motifs suivants : S, SW, SWH, R, RN et SWN?

1. On ne doit pas avoir un même motif X étiqueté '+' et '-' dans la relation.

RowId	Item	Classe
1	S W N	'+'
2	S W H	'+'
3	R C H	'-'

2.2 Classification Bayesienne Simple

Ce modèle de classification est très simple à mettre en œuvre. Il est aussi le plus utilisé dans la pratique car il offre très bon rapport « *Simplicité / Efficacité* ». Il s'appuie sur une version simplifiée de la formule de Bayes en supposant qu'il y a indépendance entre les règles d'association, appelées règles de classification dans ce contexte. Il permet aussi la prise en compte de plusieurs valeurs pour l'attribut Classe et non plus deux comme dans l'espace de versions.

2.2.1 Le classeur

Il est composé des règles unaires $c_i \Rightarrow a$ ($\forall a \in \mathcal{I}$ et $c_i \in \mathcal{C}$) munies de leur confiance ainsi que de la fréquence de chaque classe c_i.

2.2.2 Le classifieur

Pour chaque classe c_i, nous allons évaluer la confiance de la règle $X \Rightarrow c_i$ en utilisant la formule suivante :

$$Conf(X \Rightarrow c_i) = Freq(c_i) \prod_{a \in X} Conf(c_i \Rightarrow a).$$

La classe choisie est celle pour laquelle la confiance est maximale.

Exemple 2.3 Soit r_2 la relation suivante :

Le classeur obtenu est le suivant :

RowId	Item	Classe
1	ABC	c_1
2	DE	c_1
3	ABE	c_1
4	ABC	c_2
5	BC	c_3
6	ADE	c_3

Règle	Confiance
$c_1 \Rightarrow A$	2/3
$c_1 \Rightarrow B$	2/3
$c_1 \Rightarrow C$	1/3
$c_1 \Rightarrow D$	1/3
$c_1 \Rightarrow E$	1/3
$c_2 \Rightarrow A$	1
$c_2 \Rightarrow B$	1
$c_2 \Rightarrow C$	1
$c_3 \Rightarrow A$	1/2
$c_3 \Rightarrow B$	1/2
$c_3 \Rightarrow C$	1/2
$c_3 \Rightarrow D$	1/2
$c_3 \Rightarrow E$	1/2

TABLE 2.2: Classeur pour la relation r_2

1. Regardons à quelle classe appartient le motif BC.

   - $conf(BC \Rightarrow c_1) = Freq(c_1) * *conf(c_1 \Rightarrow B) * conf(c_1 \Rightarrow C) = 1/2 * 2/3 * 1/3 = 1/9$.
   - $conf(BC \Rightarrow c_2) = Freq(c_2) * conf(c_2 \Rightarrow B) * conf(c_2 \Rightarrow C) = 1/6 * 1 * 1 = 1/6$.
   - $conf(BC \Rightarrow c_3) = Freq(c_3) * conf(c_3 \Rightarrow B) * conf(c_3 \Rightarrow C) = 1/3 * 1/2 * 1/2 = 1/12$.

Donc $Classe(BC) = c_2$.

2. Regardons à quelle classe appartient le motif D.

   - $conf(D \Rightarrow c_1) = Freq(c_1) * conf(c_1 \Rightarrow D) = 1/2 * 1/3 = 1/6$.
   - $conf(D \Rightarrow c_2) = Freq(c_2) * conf(c_2 \Rightarrow D) = 1/6 * 0$.
   - $conf(D \Rightarrow c_3) = Freq(c_3) * conf(c_3 \Rightarrow D) = 1/3 * 1/2 = 1/6$.

   Nous ne pouvons pas déterminer à quelle classe va appartenir le motif D car $conf(D \Rightarrow c_1) = conf(D \Rightarrow c_3)$. On admet donc que D puisse avoir plusieurs classes : c_1 et c_3.

Exercice 2.2
Reprenez la relation de l'exercice précédent et calculez son classeur associé. Puis regardez comment sont classés les motifs S, SW, SWH, R, RN et SWN. Que constatez-vous?

II

Fouille de base de données n-aires

Chapitre

3

Sommaire

3.1 Rappels sur les dépendances fonctionnelles
3.2 Ensembles maximaux et dépendances fonctionnelles
3.3 Inférence des dépendances fonctionnelles
3.4 Relation d'Armstrong
3.5 Clés minimales d'une relation
3.6 Tests de formes normales

Dépendances Fonctionnelles

De par leur importance dans la conception des bases de données, dans l'analyse des bases de données existantes et dans l'optimisation de requêtes, les dépendances fonctionnelles sont un outil que les administrateurs et concepteurs de bases de données se doivent de connaître et de maîtriser. Nous verrons comment nous pouvons calculer un ensemble de dépendances fonctionnelles à partir d'un autre ensemble de dépendances fonctionnelles puis à partir d'une relation. Nous verrons ensuite comment nous pouvons calculer une relation exemple r' d'une relation r, r' ne devant satisfaire que les dépendances fonctionnelles valides sur r. De plus, nous essayerons de minimiser la taille de r'. Cette approche se justifie aisément car il est plus facile de manipuler une petite relation plutôt qu'une relation pouvant attendre plusieurs giga-octets de données (la base de données de France Telecom occupe plus de 29.232 Go) ou plusieurs milliards de tuples (la base de données de AT&T contient plus de 496 milliards de tuples) [1]. Puis nous verrons comment nous pouvons obtenir les clés (primaires) d'une relation. Pour finir, nous testerons si les schémas relationnels

1. source : http://www.wintercorp.com/vldb/2003_TopTen_Survey/TopTenWinners.asp

d'une relation sont en troisième forme normale ou en forme normale de Boyce-Codd. [2]

3.1 Rappels sur les dépendances fonctionnelles

3.1.1 Un peu de vocabulaire

Un **schéma de base de données** [3] est un ensemble de schémas de relation. Un **schéma de relation** est un ensemble d'attributs et un **attribut** prend ses valeurs dans un **domaine**. Une **BD** est un ensemble de relations et une **relation** est un ensemble de tuples sur un schéma de relation.

Soit $X \subseteq \mathcal{R}$, $\mathcal{R}$ un schéma de relation et t un tuple, $t[X]$ est la **restriction** de t à X. Toute relation possède un attribut $RowId$ qui est un identifiant unique pour chaque tuple. $RowId$ est présent dans tous les SGBD-R (Système de Gestion de Bases de Données Relationnelles) et peut servir de clé primaire pour toute relation si aucune clé n'est spécifiée par le concepteur du schéma (au même titre que tous les attributs du schéma). Cependant, nous n'utiliserons cet attribut que pour numéroter les tuples et non comme attribut faisant partie du schéma relationnel.

Exemple 3.1 Soit r :

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_1	b_3	c_1

$\Rightarrow$

RowId	A	B	C
1	a_1	b_1	c_1
2	a_1	b_2	c_2
3	a_1	b_3	c_1

On dit que r est une relation sur $\mathcal{R} = \{A, B, C\}$ ou que $\mathcal{R}$ est le schéma de la relation r.

Une **dépendance fonctionnelle** [4] sur $\mathcal{R}$ est une expression de la forme $X \rightarrow Y, (X, Y) \subseteq \mathcal{R}^2$. X est la partie gauche ou *lhs* (left hand side) de la DF et Y est sa partie droite ou *rhs* (right hand side).

2. Le lecteur intéressé pourra lire [18].
3. noté BD par la suite.
4. notée DF par la suite.

Exemple 3.2 Sur la BD exemple 3.1, $A \rightarrow B$ et $BC \rightarrow A$ sont deux DF.

Une DF $X \rightarrow Y$ est **valide** sur r (noté $r \vDash X \rightarrow Y$) si et seulement si $\forall t_i, t_j \in r$, si $t_i[X] = t_i[Y]$ alors $t_j[X] = t_j[Y]$. Autrement dit, étant donné une DF $X \rightarrow Y$ et deux tuples t_i, t_j, si ces deux tuples ont même valeur (attribut par attribut) sur X alors ils ont même valeur sur Y. Une DF $X \rightarrow Y$ est **triviale** si $Y \subseteq X$. Une DF $X \rightarrow Y$ est **minimale** si $\nexists Z \subset X$ tel que $Z \rightarrow Y$. Une DF est **standard** si $X \neq \emptyset$. Dans la suite de ce document, nous considérerons que toutes les DF sont sous forme standard.

Exemple 3.3 Sur la BD exemple 3.1, la DF $A \rightarrow B$ n'est pas valide sur r (les deux premiers tuples la contredise) alors que la DF $BC \rightarrow A$ est valide. De plus cette DF est non triviale et non minimale car la DF $C \rightarrow A$ est valide sur r. Les 3 DF données jusqu'à présent dans ce chapitre sont des DF standard.

Exercice 3.1
Dans quel cas a-t-on une DF standard (*i.e.* $\emptyset \rightarrow X, X \in \mathcal{R}$)?

Les **axiomes d'Armstrong** définissent des règles que les DF doivent respecter et qui nous permettent de construire d'autres DF.

1. si $Y \subseteq X$, alors $X \rightarrow Y$ (*reflexivité*)
2. si $X \rightarrow Y$, alors $XZ \rightarrow YZ$ (*augmentation*)
3. si $X \rightarrow Y$ et $Y \rightarrow Z$, alors $X \rightarrow Z$ (*transitivité*)
4. si $X \rightarrow Y$ et $YZ \rightarrow W$, alors $XZ \rightarrow W$ (*pseudo-transitivité*)

Soit $\mathcal{F}$ un ensemble de DF valides sur r, alors $\mathcal{F} \vDash X \rightarrow Y$ si $X \rightarrow Y$ peut être déduite de $\mathcal{F}$ à partir de (1), (2), (3) et (4).

Exemple 3.4 Sur la BD exemple 3.1, considérons $\mathcal{F} = \{B \rightarrow C, C \rightarrow A\}$, en utilisant la transitivité nous obtenons la DF $B \rightarrow A$. Par conséquent, nous pouvons écrire $\mathcal{F} \vDash B \rightarrow A$. De plus nous avons aussi $BC \rightarrow B$ (reflexivité) et $B \rightarrow BC$ (augmentation).

Remarque : Attention, ce n'est pas parce que la DF $X \rightarrow Y$ est valide que la DF $Y \rightarrow X$ est valide aussi!

3.1.2 Dépendances fonctionnelles, bases et couvertures

L'ensemble de toutes les DF valides, $\mathcal{F}^+ = \{X \to Y : \mathcal{F} \vDash X \to Y\}$, est la **fermeture de** $\mathcal{F}$.

Exemple 3.5 Considérons l'ensemble $\mathcal{F} = \{B \to C, C \to A, B \to A\}$, alors $\mathcal{F}^+ = \{B \to C, C \to A, B \to A, BC \to AC, BC \to AB\}$

Définition 3.1 Couverture
Soient $\mathcal{F}$ et $\mathcal{G}$ deux ensembles de DF. $\mathcal{F}$ est une couverture de $\mathcal{G}$ si et seulement si $\mathcal{F}^+ = \mathcal{G}^+$.

Exemple 3.6 En considérant l'ensemble de DF $\mathcal{F}$ donné dans l'exemple 3.5, l'ensemble de DF $\mathcal{G} = \{B \to C, C \to A\}$ est un couverture de $\mathcal{F}$ car nous pouvons retrouver la DF $B \to A$ par transitivité.

Exercice 3.2
Trouver une autre couverture que $\mathcal{G}$ pour $\mathcal{F}$.

Définition 3.2 Base canonique
Une couverture $\mathcal{F}$ est dite canonique si $\forall X \to Y \in \mathcal{F}$ on a :

- $|Y| = 1$
- $X \to Y$ est minimale

On dit aussi que $\mathcal{F}$ est une base canonique.

Exemple 3.7 En considérant l'ensemble de DF $\mathcal{F}$ donné dans l'exemple 3.5 et l'ensemble de DF $\mathcal{G}$ donné dans l'exemple 3.6, alors $\mathcal{F}$ et $\mathcal{G}$ sont deux bases canoniques de $\mathcal{F}^+$.

Exercice 3.3
Est-ce que la couverture que vous avez trouvée dans l'exercice précédent est une base canonique pour $\mathcal{F}^+$?

Puisque la base canonique n'est pas unique, il peut être intéressant de ne garder que certaines DF tout en conservant la propriété de couverture.

Définition 3.3 Base non redondante/minimale
Une couverture $\mathcal{F}$ est non redondante si et seulement si $\nexists \mathcal{G} \subset F$ tel que $\mathcal{F}^+ = \mathcal{G}^+$. Une couverture est minimale si elle est canonique et non redondante.

Exemple 3.8 L'ensemble de DF $\mathcal{F}$ donné dans l'exemple 3.5 est une base minimale pour $\mathcal{F}^+$.

3.1.3 Dépendances fonctionnelles et fermeture

Étant donné un ensemble de DF $\mathcal{F}$, il existe un opérateur de fermeture sur des attributs de $\mathcal{R}$, mais contrairement au chapitre **??**, l'opérateur de fermeture n'est pas l'intersection. En effet quelle sémantique pourrait avoir $X \rightarrow Y \cap X' \rightarrow Y'$?

Définition 3.4 Opérateur de fermeture, fermé et clé
Soit $\mathcal{F}$ un ensemble de DF valides sur $\mathcal{R}$, alors l'application :

$$^+ : \mathcal{P}(\mathcal{R}) \rightarrow \mathcal{P}(\mathcal{R})$$
$$X \mapsto X \cup \max_{\subseteq}(\{Y \subseteq \mathcal{R} : \mathcal{F} \vDash X \rightarrow Y\})$$

est un opérateur de fermeture. Par analogie avec le chapitre **??**, si $X^+ = X$, alors X est un fermé. On note par $CL(\mathcal{R})$ les sous-ensembles de $\mathcal{R}$ qui sont des fermés. La définition d'une clé (Cf. definition 5.16 p. 90) reste la même, seul l'opérateur de fermeture change. De plus, si X est une clé pour $\mathcal{R}$ alors X est une **clé minimale pour la relation** r, $\mathcal{R}$ étant le schéma de r.

Exemple 3.9 Soit $\mathcal{F} = \{AB \rightarrow D, BC \rightarrow D, D \rightarrow E, E \rightarrow A\}$, alors $E^+ = AE$ et $BC^+ = ABCDE$.

Cependant obtenir des fermés avec la définition précédente reste assez laborieux, c'est pourquoi nous allons utiliser l'algorithme Closure pour calculer les fermés.

Alg. 8 Closure

Entrée : Ensemble de DF $\mathcal{F}$ valides sur $\mathcal{R}$, $X \subseteq \mathcal{R}$
Sortie : X^+

1: $result \leftarrow X$
2: **tant que** $old \neq result$ **faire**
3: $\quad old \leftarrow result$
4: $\quad$ **pour tout** $Y \rightarrow Z \in \mathcal{F}$ **faire**
5: $\quad\quad$ **si** $Y \subseteq result$ et $Z \nsubseteq result$ **alors** $result = result \cup Z$
6: $\quad$ **fin pour**
7: **fin tant que**

Exemple 3.10 Exécutons cet algorithme afin de nous assurer que nous avons obtenu «*les bons fermés*» lors de l'exemple précédent. (le symbole «...» désigne une DF quelconque autre que celle qui ont déjà été exprimées).

1. Calculons E^+ :

DF	*old*	*result*
		E
	E	E
$AB \rightarrow D$	E	E
...	E	E
$E \rightarrow A$	E	EC
	AE	AE
...	AE	AE

Donc nous avons bien $E^+ = AE$.

2. Calculons BC^+ :

DF	*old*	*result*
		BC
	BC	BC
$AB \rightarrow D$	BC	BC
$BC \rightarrow D$	BC	BCD
	BCD	BCD
$D \rightarrow E$	BCD	$BCDE$
	$BCDE$	$BCDE$
$E \rightarrow A$	$BCDE$	$ABCDE$
	$ABCDE$	$ABCDE$
...	$ABCDE$	$ABCDE$

Donc nous avons bien $BC^+ = ABCDE$.

Il peut être nécessaire de devoir calculer l'ensemble des fermés de $\mathcal{R}$ sur $\mathcal{F}$. Cependant une exploration exhaustive du treillis des parties de $\mathcal{R}$ (espace de recherche) n'est pas une bonne solution. En effet si nous avons beaucoup d'attributs (e.g. $|\mathcal{R}| = 20$), alors nous devons explorer un nombre **exponentiel** de solutions (i.e. $|\mathcal{P}(\mathcal{R})| = 2^{20} \simeq 1E7$). Nous allons utiliser l'algorithme Next-Closure (Cf. algorithme 16 p. 95) pour calculer l'ensemble des fermés. Il nous suffit juste de remplacer l'opérateur de fermeture h par $^+$ que nous venons de définir. Il est à souligner que Next-Closure a l'agréable propriété de retourner l'ensemble des fermés quel que soit l'opérateur de fermeture qui lui est donné en paramètre.

Exemple 3.11 La trace de l'algorithme Next-Closure en utilisant l'ensemble de DF de l'exemple 3.9 est donnée dans le tableau 3.1. De plus, nous avons $CL(\mathcal{R}) = \{\emptyset, A, B, C, AC, AE, ACE, ADE, ABDE, ACDE, \mathcal{R}\}$.

En organisant l'ensemble des fermés avec l'ordre d'inclusion et en utilisant le théorème de Birkhoff (Cf. théorème 5.2 p. 89), alors nous sommes capables de construire un treillis appelé, lui aussi, **treillis des fermés** (ou treillis de Galois). De plus les opérateur Meet ($\bigwedge$) et Join ($\bigvee$) du treillis s'exprime de la manière suivante :

Theorème 3.1

$$\forall X_1, .., X_n \subseteq \mathcal{P}(\mathcal{R}), \bigvee_i X_i = (\bigcup_i X_i)^+ \text{ et } \bigwedge_i X_i = \bigcap_i X_i.$$

X	S	i	$Inc(X,i)$	X^*	$T = max(X^* \backslash X)$	lower
$\emptyset$	$ABCDE$	A	A	A	$\emptyset$	oui
A	$BCDE$	B	B	B	$\emptyset$	oui
B	$ACDE$	A	AB	ABD	D	non
		C	C	C	C	oui
C	$ABDE$	A	AC	AC	A	oui
AC	BDE	B	BC	$ABCDE$	E	non
		D	D	ADE	E	non
		E	E	AE	E	oui
AE	BCD	B	BE	$ABDE$	D	non
		C	CE	ACE	C	oui
ACE	BD	B	BCE	$ABCDE$	D	non
		D	DE	ADE	D	oui
ADE	BC	B	BDE	$ABDE$	B	oui
$ABDE$	C	C	CDE	$ACDE$	C	oui
$ACDE$	B	B	$BCDE$	$ABCDE$	B	oui
$ABCDE$						

TABLE 3.1: Ensemble des fermés obtenus avec Next-Closure

Exemple 3.12 Puisque nous avons un treillis, nous sommes donc capable de dessiner son diagramme de Hasse. Voici le diagramme de Hasse du treillis des fermés relatif à l'ensemble de DF $\mathcal{F}$ sur $\mathcal{R}$ de l'exemple 3.9.

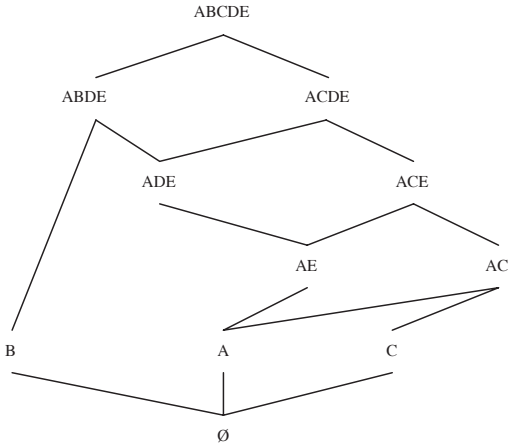


FIGURE 3.1: Treillis de Galois de $\mathcal{R}$ sur $\mathcal{F}$

Exercice 3.4
Soit l'ensemble de DF suivant : $\mathcal{F}_1 = \{D \rightarrow AC, B \rightarrow E, E \rightarrow B, BC \rightarrow E, CE \rightarrow B, AB \rightarrow CE, AE \rightarrow BC, A \rightarrow C\}$ valide sur $\mathcal{R} = ABCDE$. Calculer $CL(\mathcal{R})$. Comparer cette exécution de Next-Closure avec celle page 95. Que remarquez vous?

En conséquence du théorème précédent, nous n'avons pas besoin de tous les fermés pour connaître $CL(\mathcal{R})$. Seuls les éléments meet-irréductibles (un seul arc sortant) sont importants. Les autres peuvent se retrouver par intersection(s) des meet-irréductibles.

Définition 3.5 Famille génératrice pour un système de fermeture
On appelle famille génératrice pour un système de fermeture stable par intersection(s) l'ensemble des fermés qu'on ne peut obtenir par intersection d'autres fermés. Cette ensemble de fermés est exactement l'ensemble des meet-irréductibles

du système de fermeture $CL(\mathcal{R})$. Cette famille, notée $GEN(\mathcal{F})$ peut être définie de la façon suivante : $GEN(\mathcal{F}) = \{X \in CL(\mathcal{R} \mid X^+_{CL(\mathcal{R})\backslash X} \neq X\}$.
Autrement dit, $X \in CL(\mathcal{R})$ est un meet-irréductible si et seulement si on ne peut pas le retrouver par intersection(s) successive(s) d'éléments appartenant à $CL(\mathcal{R})\backslash X$.

Exemple 3.13 Avec l'ensemble de fermés obtenu précédemment, nous avons : $GEN(\mathcal{F}) = \{B, C, AC, ACE, ABDE, ACDE\}$. De plus, $ADE \notin GEN(\mathcal{F})$ car $ADE = ABDE \cap ACDE$.

Exercice 3.5
En considérant l'exercice 3.4, quelle est la famille génératrice pour l'ensemble des fermés?

Cependant la méthode utilisée n'est «*pas très efficace*» car pour chaque fermé X il nous faut recalculer sa fermeture à partir de $CL(\mathcal{R})\backslash X$ pour savoir si $X \in GEN(\mathcal{F})$. Donc, dans le pire des cas, $2^{|\mathcal{R}|}$ fermés devront être recalculés. Ce cas «*extreme*» est obtenu lorsque chaque sous-ensemble de $\mathcal{R}$ est un fermé et donc $CL(\mathcal{R}) = \mathcal{P}(\mathcal{R})$. Nous allons donc utiliser une autre caractérisation de $GEN(\mathcal{F})$ pour retrouver cet ensemble rapidement.

3.2 Ensembles maximaux et dépendances fonctionnelles

3.2.1 Les ensembles maximaux

C'est une notion utile pour caractériser les DF et beaucoup d'autres problèmes similaires comme les tests de forme normale et les relations d'Armstrong, les DF approximatives, ...

Définition 3.6 Ensembles maximaux
Soit $\mathcal{R}$ un schéma de relation, $\mathcal{F}$ un ensemble de DF valides sur $\mathcal{R}$ et $A \in \mathcal{R}$ un attribut. Les ensembles maximaux de A par rapport à $\mathcal{F}$, notés $max(\mathcal{F}, A)$ sont définis par : $max(\mathcal{F}, A) = \{X \subseteq \mathcal{R} \mid \mathcal{F} \not\models X \rightarrow A$ et $\forall B \in \mathcal{R}\backslash X, \mathcal{F} \models XB \rightarrow A\}$. Si l'ensemble de DF $\mathcal{F}$ est explicite dans son contexte, alors nous noterons $max(A)$ à la place de $max(\mathcal{F}, A)$. De plus, $max(\mathcal{F})$ regroupe tous les maximaux des tous les attributs : $max(\mathcal{F}) = \bigcup_{A \in \mathcal{R}} max(\mathcal{F}, A)$.

Cependant, cette définition des ensembles maximaux n'est pas aisée à mettre en œuvre c'est pourquoi nous allons utiliser la caractérisation suivante :

Theorème 3.2
Soit $\mathcal{R}$ un schéma de relation, $\mathcal{F}$ un ensemble de DF valides sur $\mathcal{R}$ et $A \in \mathcal{R}$ un attribut. Alors nous avons : $max(\mathcal{F}, A) = \max_{\subseteq}\{Y \in CL(R) \mid A \notin Y\}$.

Exemple 3.14 En considérant l'ensemble de DF $\mathcal{F}$ donné dans l'exemple 3.9, nous avons :

- $max(A) = \{B, C\}$,
- $max(B) = \{ACDE\}$,
- $max(C) = \{ABDE\}$,
- $max(D) = \{ACE, B\}$,
- $max(E) = \{AC, B\}$.

De plus, nous avons $max(\mathcal{F}) = \{B, C, AC, ACE, ABDE, ACDE\}$.

Exercice 3.6
En considérant l'exercice 3.4, calculer $max(\mathcal{F})$.

Nous pouvons faire le lien entre les ensembles maximaux et la famille génératrice de la manière suivante :

Theorème 3.3
Soit $\mathcal{R}$ un schéma de relation, $\mathcal{F}$ un ensemble de DF valides sur $\mathcal{R}$. Alors nous avons $max(\mathcal{F}) = GEN(\mathcal{F})$.

Donc, si nous devons calculer la famille génératrice d'un système de fermeture, avec la solution qui nous était proposée dans la partie concernant les rappels, nous faisons $|CL(\mathcal{R})| * \textit{coût de calcul d'un fermé} \simeq o(|\mathcal{R}|^2 * |CL(\mathcal{R})|^2)$ opérations. Avec cette deuxième caractérisation nous faisons aussi $o(|\mathcal{R}|^2 * |CL(\mathcal{R})|^2)$ opérations. Donc la complexité théorique des deux approches sont identiques, cependant, dans la pratique, la deuxième approche s'avère plus rapide que la première...

3.2.2 Transversaux d'un hypergraphe

Puisque nous cherchons des DF minimales du type $X \rightarrow A$, Il est possible de faire le lien entre X et le complémentaire de $max(A)$ dans $\mathcal{R}$ en utilisant les transversaux minimaux. Nous allons définir ce concept et allons voir comment nous pouvons les calculer?

Définition 3.7 Transversal (d'un hypergraphe)
Soit $\mathcal{R}$ un ensemble d'attributs, E un ensemble d'ensembles tel que $E \subseteq \mathcal{P}(\mathcal{R})$ et $X \subseteq E$. Alors :

- X est un transversal de E si et seulement si $X \cap Y \neq \emptyset, \forall\ Y \in E$,
- X est un transversal de $\overline{E} = \{\mathcal{R} \backslash X \mid X \in E\}$ si et seulement si $X \nsubseteq Y, \forall\ Y \in E$.

Cette notion est connue depuis longtemps en «*théorie des graphes*», en fait l'ensemble d'ensembles E est un $|\mathcal{R}|$-hypergraphe. Si $|\mathcal{R}| = 2$, alors l'hypergraphe est appelé graphe.

Remarque : Avec cette definition des transversaux de $\overline{E}$, nous n'avons pas besoin de connaître explicitement $\overline{E}$ car il n'intervient pas dans cette définition. Bien sûr nous avons la définition suivante pour les transversaux de $\overline{E}$: X est un transversal de $\overline{E}$ si et seulement si $X \cap Y \neq \emptyset, \forall\ Y \in \overline{E}$.

La contrainte «*X est un transversal de E (ou de $\overline{E}$)*» est une contrainte monotone selon l'ordre d'inclusion. En conséquence de la propriété 5.2 (p. 84), la connaissance des minimaux transversaux fournit la connaissance de tous les autres transversaux. Ainsi, tout sur-ensemble d'un transversal est, lui aussi, un transversal.

Exemple 3.15 Soit $\mathcal{R} = ABCDE$ et $E = \{D, AB, BC, BE\}$. BD, ABD, BCD, ... sont des transversaux de E. BD est un transversal minimal. $AD, ABD, ...$ sont des transversaux de $\overline{E}$.

3.2.3 Découverte des minimaux transversaux

Soit $\mathcal{R}$ un ensemble d'attributs, E un ensemble d'ensembles tel que $E \subseteq \mathcal{P}(\mathcal{R})$. On note $Tr(E)$ l'ensemble de minimaux transversaux de E à valeur dans $\mathcal{R}$

et $Tr(\overline{E})$ l'ensemble de minimaux transversaux de $\overline{E}$ à valeur dans $\mathcal{R}$. Le problème «*trouver tous les transversaux de E (ou de $\overline{E}$) à valeur dans $\mathcal{R}$*» appartient à la famille des problèmes NP-Complet. Ceci implique qu'il n'existe pas d'algorithme polynomial pour résoudre ce problème. Nous allons utiliser un algorithme semblable à celui de Norris en construisant une suite d'unions. Au début de l'algorithme, la suite est initialisée à l'élément vide ($\emptyset$). Puis, pour chaque élément X dans E, nous construisons la suite de la manière suivante : $S := \min_{\subseteq}(\{Y \cup x \mid Y \in S \text{ et } x \in X\})$ si nous cherchons les minimaux transversaux de E. Si nous cherchons les minimaux transversaux de $\overline{E}$, alors nous devons construire cette suite : $S := \min_{\subseteq}(\{Y \cup x \mid Y \in S \text{ et } x \in \mathcal{R} \backslash X\})$. Ensuite, nous répétons ce processus jusqu'à ce que tous les éléments de E aient été examinés.

Alg. 9 TR (suite)

Entrée : $\mathcal{R}$ un ensemble d'attributs et $\mathcal{E}$ un ensemble d'ensembles à valeur dans $\mathcal{R}$
Sortie : $Tr(E)$ ou $Tr(\overline{E})$
1: $S = \emptyset$
2: **pour tout** $X \in E$ **faire**
3: $\quad S := \min_{\subseteq}(\{Y \cup x \mid Y \in S \text{ et } x \in X\}) \backslash\backslash x \in \mathcal{R} \backslash X$ si on cherche $Tr(\overline{E})$
4: **fin pour**
5: **return** S

Exemple 3.16 Soit $\mathcal{R} = ABCDE$ et $E = \{AB, BC, BE, D\}$. Regardons la trace de l'algorithme TR dans le cas d'une recherche des minimaux transversaux.

- Étape 0 :
  $S = \{\emptyset\}$
- Étape 1 :
  $S = \min_{\subseteq}\{\emptyset\}^{AB} = \min_{\subseteq}\{A, B\} = \{A, B\}$.

  Remarquons la notation utilisée $\min_{\subseteq}\{S\}^{X}$. Afin d'obtenir plus de facilité pour exécuter l'algorithme, l'élément X courant est mis en exposant. Ainsi, nous pouvons visualiser directement quels sont les attributs $x \in X$.

- Étape 2 :
  $\overline{S = \min_{\subseteq}\{A, B\}^{BC}} = \min_{\subseteq}\{AB, AC, B, BC\} = \{B, AC\}$.
- Étape 3 :
  $\overline{S = \min_{\subseteq}\{B, AC\}^{BE}} = \min_{\subseteq}\{B, BE, ABC, ACE\} = \{B, ACE\}$.
- Étape 4 :
  $\overline{S = \min_{\subseteq}\{B, ACE\}^{D}} = \min_{\subseteq}\{BD, ACDE\} = \{BD, ACDE\}$

Par conséquent, $Tr(E) = \{BD, ACDE\}$.

3.2.4 Des ensembles maximaux aux dépendances fonctionnelles

Nous allons utiliser les ensembles maximaux pour obtenir une couverture canonique de l'ensemble de DF. Nous allons définir l'opérateur lhs afin d'obtenir des DF du type $lhs(A) \rightarrow A$. Ensuite nous ferons le lien entre les parties gauches des DF ayant pour partie droite A et $max(A)$ en utilisant les transversaux minimaux du complémentaire de $max(A)$.

Définition 3.8 Opérateur Left hand side
Soit $\mathcal{R}$ un schéma de relation, $\mathcal{F}$ un ensemble de DF valides sur $\mathcal{R}$ et $A \in \mathcal{R}$ un attribut. On définit un opérateur left hand side (lhs) comme suit :
$lhs(A) = \{X \subseteq \mathcal{R} \mid \mathcal{F} \vDash X \rightarrow A, \nexists Y \subset X \text{ tel que } \mathcal{F} \vDash X \rightarrow A\}$.

Exemple 3.17 En considérant l'ensemble de DF $\mathcal{F}$ donné dans l'exemple 3.9, nous avons :

- $lhs(A) = \{A, D, E, BC\}$
- $lhs(B) = \{B\}$
- $lhs(C) = \{C\}$
- $lhs(D) = \{D, AB, BC, BE\}$
- $lhs(E) = \{D, E, AB, BC\}$

Définition 3.9 Complémentaire aux ensembles maximaux
Soit $\mathcal{R}$ un schéma de relation, $\mathcal{F}$ un ensemble de DF valides sur $\mathcal{R}$, $A \in \mathcal{R}$ un attribut et $max(A)$ ses maximaux (Cf. definition 3.6). On peut définir le complémentaire aux ensembles maximaux pour un attribut A, noté $cmax(A)$, comme suit : $cmax(A) = \{\overline{X} \mid X \in max(A)\} = \{\mathcal{R} \backslash X \mid X \in max(A)\}$.

Exemple 3.18 En considérant l'ensemble de DF $\mathcal{F}$ donné dans l'exemple 3.9, nous avons :

- $cmax(A) = \{ACDE, ABDE\}$,
- $cmax(B) = \{B\}$,
- $cmax(C) = \{C\}$,
- $cmax(D) = \{BD, ACDE\}$,
- $cmax(E) = \{BDE, ACDE\}$.

Le théorème suivant fournit le lien entre le complémentaire de maximaux d'un attribut A et les parties gauches des DF ayant pour partie droite A.

Theorème 3.4
Soit $\mathcal{R}$ un schéma de relation, $\mathcal{F}$ un ensemble de DF valides sur $\mathcal{R}$ et $A \in \mathcal{R}$ un attribut, alors les trois assertions suivantes sont équivalentes :

1. $lhs(A) = Tr(cmax(A))$
2. $lhs(A) = Tr(\overline{max(A)})$
3. $cmax(A) = Tr(lhs(A))$

Exemple 3.19 En considérant l'ensemble de DF $\mathcal{F}$ donné dans l'exemple 3.9, nous avons pour l'attribut A : $lhs(A) = \{A, D, E, BC\}$. Ainsi les DF suivantes appartiennent à la base canonique de $\mathcal{F}$: $D \rightarrow A, E \rightarrow A, BC \rightarrow A$. La DF $A \rightarrow A$ étant toujours vérifiée, elle est omise dans la base canonique.

En appliquant ce théorème à chaque attribut de $\mathcal{R}$, nous pouvons ainsi construire la base canonique de $\mathcal{F}$. C'est que ce que fait l'algorithme suivant :

Alg. 10 Base canonique (v1)

Entrée : $\mathcal{R}$ un ensemble d'attributs et $\mathcal{F}$ un ensemble de DF valides sur $\mathcal{R}$
Sortie : $\mathcal{F}'$ base canonique de $\mathcal{F}$
1: $CL(\mathcal{R}) = Next - Closure(\mathcal{R}, \mathcal{F})$
2: **pour tout** $A \in \mathcal{R}$ **faire**
3: calculer $cmax(A)$
4: calculer $Tr(cmax(A)$
5: $\mathcal{F}' = \mathcal{F}' \cup \{X \rightarrow A\} \mid X \in Tr(cmax(A))$
6: **fin pour**
7: **return** $\mathcal{F}'$

Exemple 3.20 En considérant l'ensemble de DF $\mathcal{F}$ donné dans l'exemple 3.9, nous avons :

item	$cmax$	lhs
A	$ACDE, ABDE$	A, D, E, BC
B	B	B
C	C	C
D	$BD, ACDE$	D, AB, BC, BE
E	$BDE, ACDE$	D, E, AB, BC

Ainsi la base canonique de $\mathcal{F}$ est $\{D \rightarrow A, E \rightarrow A, BC \rightarrow A, AB \rightarrow D, BC \rightarrow D, BE \rightarrow D, D \rightarrow E, AB \rightarrow E, BC \rightarrow E\}$.

Exercice 3.7
En considérant l'exercice 3.4, calculer la base canonique de $\mathcal{F}$.

3.3 Inférence des dépendances fonctionnelles

Nous venons de voir comment générer une base canonique à partir d'un ensemble de DF. Mais, lorsqu'on examine une base de donnée via un SGBD-R (Système de Gestion de Bases de Données Relationnelles), on peut se demander quelles sont les DF qui sont vérifiées sur cette base. Une approche naïve consisterait à tester toutes les DF possibles. Ce n'est bien sûr pas cette approche que nous allons suivre. Nous allons nous appuyer sur le concept des ensembles en accord sur la relation et faire le lien entre ceux-ci et les maximaux. Ainsi, seule

la première phase de l'algorithme précédent changera et nous pourrons extraire de la relation un ensemble de DF sous forme canonique.

Définition 3.10 Ensembles en accord
Soit r une relation de schéma $\mathcal{R}$, t_1 et t_2 deux tuples, $X \subseteq \mathcal{R}$. Nous disons que t_1 et t_2 sont en accord sur X si et seulement si $t_1[X] = t_2[X]$. L'ensemble en accord de t_1 et t_2, noté $ag(t_1, t_2)$, est définit par : $ag(t_1, t_2) = \{X \in \mathcal{R} \mid t_1[X] = t_2[X]\}$. On peut étendre cette définition à la relation elle-même comme ceci : $ag(r) = \{ag(t_1, t_2), t_1 \neq t_2\}$.

Exemple 3.21 Soit r la BD de schéma $\mathcal{R} = ABCDE$ suivante :

A	B	C	D	E
0	0	0	0	0
O	7	0	7	7
1	0	1	1	1
0	0	5	0	0
0	3	0	3	3
0	6	0	0	0
2	2	0	2	2
0	4	0	4	0

TABLE 3.2: Relation exemple r_1

Nous avons $ag(t_3, t_7) = AE$. De plus, $ag(r) = \{\emptyset, A, B, C, AC, AE, ACE, ADE, ABD$

Remarque sur les relations : Dans la suite de ce paragraphe, nous considérerons que tous les attributs ont pour domaine celui des entiers. Pour les autres types de domaines, il suffit de faire correspondre chaque valeur à un entier unique.

Exercice 3.8
Calculer $ag(r)$ pour la relation suivante :

A	B	C	D	E
1	2	0	1	2
1	1	2	0	1
0	1	0	1	3
1	2	3	4	5

TABLE 3.3: Relation exemple r_2

Il nous reste maintenant à faire le lien entre les ensembles en accord de la relation et les maximaux.

Theorème 3.5
Soit r une relation de schéma $\mathcal{R}$ et $\mathcal{F}$ une couverture de l'ensemble de DF valides sur r. Alors nous avons : $GEN(\mathcal{F}) \subseteq ag(r) \subseteq CL(\mathcal{R})$. De plus, dans l'expression des maximaux (Cf. théorème 3.2), on peut remplacer $CL(\mathcal{R})$ par $ag(r)$. Donc nous pouvons redéfinir les maximaux de la manière suivante : $max(A) = \max_{\subseteq}(\{X \in ag(r) \mid A \notin X\})$.

En utilisant ce théorème, nous pouvons construire un algorithme similaire à celui générant une base canonique à partir d'un ensemble de DF en ne modifiant que la première ligne.

Alg. 11 Base canonique (v2)

Entrée : relation r de schéma $\mathcal{R}$
Sortie : $\mathcal{F}$ base canonique pour r
1: calculer $ag(r)$
2: **pour tout** $A \in \mathcal{R}$ **faire**
3: calculer $cmax(A)$
4: calculer $Tr(cmax(A)$
5: $\mathcal{F} = \mathcal{F} \cup \{X \rightarrow A\} \mid X \in Tr(cmax(A))$
6: **fin pour**
7: **return** $\mathcal{F}$

Exemple 3.22 En considérant les maximaux des DF de $\mathcal{F}$ donné dans l'exemple 3.9 et les ensembles en accord trouvés dans l'exercice ??, nous pouvons consta-

ter que ces deux ensembles sont identiques, c'est pourquoi la base canonique de l'ensemble de DF valides sur r est aussi celle donnée dans l'exemple 3.20.

Exercice 3.9
Calculer une base canonique pour la relation de l'exercice ??.

Calcul des ensembles en accord

Il est possible d'utiliser la requête SQL suivante pour calculer les ensembles en accord d'une relation r sur $\mathcal{R} = \{A, B, ...Z\}$:

SELECT DISTINCT
DECODE (R1.A,R2.A,'A') ||
DECODE (R1.B,R2.B,'B') ||
...
DECODE (R1.Z,R2.Z,'Z')
FROM r R1, r R2
WHERE R1.RowId < R2.RowId
AND ((R1.A = R2.A) **OR** ... **OR** (R1.Z= R2.Z))

l'opérateur DISTINCT permet de supprimer les doublons, DECODE a l'entête suivante : (arg_1, arg_2,'arg_3') et est équivalent à if $arg_1 = arg_2$ alors écrire arg_3 et || est l'opérateur de concaténation SQL.

3.4 Relation d'Armstrong

Une relation d'Armstrong est une relation qui satisfait exactement (ni plus ni moins) un ensemble de DF. Les relations d'Armstrong servent à la conception des DB. Silva *et al.* [26] ont proposé un outil dans lequel le concepteur saisit un ensemble de DF. L'outil lui présente alors une relation d'Armstrong qui est un exemple de la relation permettant de visualiser «*aisément*» les dépendances manquantes. Mannila *et al.* [19] expliquent l'utilité des relations d'Armstrong pour la conception des DB par l'exemple. Afin de calculer cette relation, nous allons nous appuyer sur les maximaux.

Construction d'une relation d'Armstrong
Soit $\mathcal{R} = X_1..X_n$ un ensemble d'attribut et $\mathcal{F}$ un ensemble de DF et $X_i \in$

$max(\mathcal{F})$. Chaque X_i est associé au tuple t_i de la manière suivante :

$$t_i[A] = \begin{cases} 0 \text{ si } A \in X_i \\ i \text{ sinon.} \end{cases}$$

Le premier tuple ne contient que des 0. La relation r ainsi créée est appelée **relation d'Armstrong**. Elle contient au moins $|max(\mathcal{F})|+1$ tuples. Cependant, la relation d'Armstrong n'a pas de taille maximale en théorie. Il suffit de considérer les maximaux déjà pris en compte et de leur associer de nouveaux tuples : i étant incrémenté à chaque fois, on ne viole aucune contrainte d'intégrité sur la relation....

Alg. 12 Construction d'une relation d'Armstrong

Entrée : $\mathcal{R}$ ensemble d'attribut, $\mathcal{F}$ ensemble de DF
Sortie : relation d'Armstrong r pour $\mathcal{F}$

```
1: calculer max(F)
2: pour tout A ∈ R faire t_0[A] = 0
3: i = 0
4: pour tout X ∈ max(F faire
5:    pour tout A ∈ R faire
6:       si A ∈ X alors t_i[A] = 0
7:                 sinon t_i[A] = i
8:    fin pour
9:    i++
10: fin pour
11: return r
```

Exemple 3.23 La relation suivante est une relation d'Armstrong pour la relation ??.

A	B	C	D	E
0	0	0	0	0
1	0	1	1	1
2	2	0	2	2
0	3	0	3	3
0	4	0	4	0
0	0	5	0	0
0	6	0	0	0

TABLE 3.4: Relation d'Armstrong pour la relation r_1

3.5 Clés minimales d'une relation

Nous allons aborder le problème de la découverte des clés minimales d'une relation. Leur découverte est très utile lors du processus de normalisation d'une relation en certaine forme normale (3NF, BCNF, ...) [10]. La connaissance des clés minimales est essentielle pour éviter certaines redondances et d'éventuelles incohérences dans la relation. Encore une fois, nous allons utiliser les maximaux et les minimaux transversaux pour faire le lien avec les clés minimales d'une relation.

Theorème 3.6
Soit r une relation de schéma $\mathcal{R}$ et $\mathcal{F}$ un ensemble de DF valides sur r. Notons $Key(r)$ les clés minimales de la relation r. Alors nous avons : $Key(r) = Tr(\overline{\max_{\subseteq}(CL(\mathcal{R})\backslash\mathcal{R})})$.

Les clés minimales d'une relation sont aussi les clés du *top* de $CL(\mathcal{R})$, soit $\mathcal{R}$ lui même. Ensuite nous avons besoin des prédécesseurs immédiats de $\mathcal{R}$ dans $CL(\mathcal{R})$. Ceci correspond à la partie $\max_{\subseteq}(CL(\mathcal{R})\backslash\mathcal{R})$ de la formule. Pour l'exemple **??**, nous obtenons $\{ABDE, ACDE\}$. Puis, il faut calculer les minimaux transversaux du complément de cet ensemble pour trouver les clé minimales de la relation. Soit, dans notre cas, $Key(r) = \{BC\}$.

Cependant, il est possible d'utiliser les maximaux et non le treillis $CL(\mathcal{R})$ dans la formule.

Corollaire 3.1 Soit r une relation de schéma $\mathcal{R}$ et $\mathcal{F}$ un ensemble de DF valides sur r. Notons $Key(r)$ les clés minimales de la relation r. Alors nous avons : $Key(r) = Tr(\overline{\max_{\subseteq}(max(\mathcal{F}))})$.

Exercice 3.10
Quelles sont les clés minimales de la relation ???

3.6 Tests de formes normales

Nous allons considérer deux problèmes qui peuvent s'exprimer ainsi : « *Étant donné un schéma relationnel $\mathcal{R}$ et un ensemble de DF $\mathcal{F}$ sur $\mathcal{R}$, $\mathcal{R}$ est-il en forme normale de Boyce-Codd ($BCNF$), ou en troisième forme normale ($3NF$), par rapport à $\mathcal{F}$?* ». Ces deux problèmes sont importants pour savoir si un schema relationnel est dans la forme voulue. Si tel n'est pas le cas, alors nous pouvons modifier le schéma en appliquant des algorithmes de normalisation [10] afin d'obtenir la forme souhaitée.

3.6.1 Test pour $BCNF$

Nous rappelons qu'un schéma $\mathcal{R}$ est en $BCNF$ si pour tout attribut $A \in \mathcal{R}$, nous avons des DF du type $X \rightarrow A \mid A \notin X$. Nous allons utiliser une caractérisation de cette forme normale en utilisant les maximaux. Pour cette caractérisation, nous avons besoin de définir les ensembles non redondants.

Définition 3.11 Ensembles non redondants
Soit un schéma relationnel $\mathcal{R}$ et un ensemble de DF $\mathcal{F}$ sur $\mathcal{R}$, un ensemble $X \subseteq \mathcal{R}$ est non redondant si et seulement si $\nexists Y \subset X \mid \mathcal{F} \vDash Y \rightarrow X$.

Le lien entre la forme normale et les maximaux est fait via le théorème suivant :

Theorème 3.7
Soit $\mathcal{F}$ un ensemble de DF sur $\mathcal{R}$. Alors $\mathcal{R}$ est en $BCNF$ si et seulement si chaque $X \in max(\mathcal{F})$ est non redondant.

Pour vérifier cette simple condition, il suffit de montrer que $\mathcal{F} \nvDash (X \backslash A) \rightarrow A$ pour chaque $A \in X$, $X \in max(\mathcal{F})$. Cette condition est équivalente à montrer que $A \notin (X \backslash A)^+$. Nous allons utiliser les maximaux pour calculer les fermés

car le treillis des fermés est un système stable par intersection (Cf. définition 3.5).

Alg. 13 Test $BCNF$

Entrée : $max(\mathcal{F}$ ensembles maximaux de l'ensemble de DF $\mathcal{F}$ sur le schéma $\mathcal{R}$

Sortie : $VRAI$ si $\mathcal{R}$ est en $BCNF$, $FAUX$ sinon

1: **pour tout** $X \in max(\mathcal{F})$ **faire**
2: **pour tout** $A \in X$ **faire**
3: **si** $A \in (X \backslash A)^+$ **alors renvoyer** $FAUX$
4: **fin pour**
5: **fin pour**
6: **return** $VRAI$

Exemple 3.24 En considérant la relation r_1 donnée dans l'exemple ??, nous voulons savoir si cette relation est en $BCNF$.

$X \in max(\mathcal{F})$	A	$(X \backslash A)^+$	$A \in (X \backslash A)^+$?
B	B	$\emptyset$	non
C	C	$\emptyset$	non
AC	A	C	non
	C	A	non
ACE	A	ACE	oui

Donc la relation r_1 n'est pas en $BCNF$.

Exercice 3.11
La relation ?? est-elle en $BCNF$?

3.6.2 Test pour $3NF$

Pour ce test, nous avons besoin d'introduire la notion d'attribut premier.

Définition 3.12 Attribut premier
Soit r une relation de schéma $\mathcal{R}$. Nous disons qu'un attribut $A \in \mathcal{R}$ est un attribut premier si et seulement si il n'existe pas une clé minimale X de $\mathcal{R}$ telle que $A \in X$. Autrement formulé, A est attribut premier si $\nexists X \in KEY(r) \mid A \in X$.

Exemple 3.25 En considérant le relation de l'exemple ??, A, D, E sont des attributs primaires.

De plus, nous rappelons qu'un schéma $\mathcal{R}$ est en $3NF$ si et seulement si pour tout attribut A non premier, nous avons $\mathcal{F} \vDash X \rightarrow A, \forall X \in \max(\mathcal{F})$ et $A \notin X$. La seule différence avec le test pour les formes en $BCNF$ réside dans ma condition appliquée aux attributs. Pour tester si une relation est en $3NF$, seuls les attributs non premiers doivent être considérés. Une telle condition peut être vérifiée à l'aide du théorème suivant :

Theorème 3.8
Soit $\mathcal{F}$ un ensemble de DF sur $\mathcal{R}$, un attribut $A \in \mathcal{R}$ est premier si et seulement si $\exists X \in max(\mathcal{F}, A) \mid (X \cup A)^+ = \mathcal{R}$.

Donc, pour tester si un attribut A est non premier, nous vérifions que $\forall X \in max(\mathcal{F}, A), (X \cup A)^+ \neq \mathcal{R}$. En apportant une petite modification à l'algorithme précédent, nous obtenons le test $3NF$.

Alg. 14 Test $3NF$

Entrée : $max(\mathcal{F}$ ensembles maximaux de l'ensemble de DF $\mathcal{F}$ sur le schéma $\mathcal{R}$
Sortie : $VRAI$ si $\mathcal{R}$ est en $3NF$, $FAUX$ sinon

```
1: pour tout X ∈ max(F) faire
2:   pour tout A ∈ X, A non premier faire
3:     si A ∈ (X\A)+ alors renvoyer FAUX
4:   fin pour
5: fin pour
6: return VRAI
```

Exemple 3.26 En considérant la relation r_1 donnée dans l'exemple ??, nous voulons savoir si cette relation est en $3NF$.

$X \in max(\mathcal{F})$	A non premier	$(X \cup A)^+$	$(X \cup A)^+ = \mathcal{R}$?
B	B	$\emptyset$	non
C	C	$\emptyset$	non
AC	C	A	non
ACE	C	ACE	oui

Donc la relation r_1 n'est pas en $3NF$.

Exercice 3.12
La relation ?? est-elle en $3NF$?

Sommaire

Chapitre

4

Conclusion

Tout au long de ce manuel, nous avons découvert différentes techniques de fouille de données. Après un rappel mathématique nécessaire, nous avons tout d'abord porté notre attention sur le modèle relationnel en caractérisant les ensembles maximaux (i) dans un premier temps à partir d'un ensemble de dépendances fonctionnelles $\mathcal{F}$, puis (ii) à partir d'une relation base de données r. Les maximaux jouent un rôle important pour la fouille de données. En effet, une fois ceux-ci calculés, nous pouvons aisément déduire une couverture canonique $\mathcal{F}'$ de l'ensemble de dépendances fonctionnelles $\mathcal{F}$, calculer les clés minimales de la relation, obtenir une relation r' de taille réduite et vérifiant uniquement les dépendances fonctionnelles valides sur r. De plus, les maximaux nous permettent de savoir, en un temps polynomial, si la relation r est en 3NF ou en BCNF. Cependant, il existe beaucoup d'autres techniques de fouille de données concernant les dépendances fonctionnelles. La base canonique n'est pas la seule couverture des dépendances fonctionnelles, la base minimale [20] peut aussi être calculée. Cette base a la propriété d'avoir un nombre minimal de dépendances fonctionnelles, mais cette base n'est pas unique pour un même ensemble de dépendances fonctionnelles : en effet la minimalité de cette base dépend de l'ordre dans lequel on considère les dépendances fonctionnelles de $\mathcal{F}$. Il existe d'autres algorithmes d'extraction des dépendances fonctionnelles, dont les plus

efficaces sont FUN [24] et $TANE$ [15]. Nous aurions aussi pu nous intéresser à la problématique de projection de dépendances fonctionnelles : « *quelles sont les dépendances fonctionnelles valides sur un sous-ensemble du schéma relationnel* » et utiliser l'algorithme RBR [12]. Une autre problématique souvent abordée est celle de la découverte des dépendances fonctionnelles approximatives : la dépendance fonctionnelle $X \Rightarrow Y$ n'est plus vérifiée sur toute la relation mais elle reste valide sur un certain nombre de tuples de la relation.

Ensuite nous avons appris les principes généraux des algorithmes par niveaux et appliqué ceux-ci pour l'extraction des minimaux transversaux et des motifs fréquents. L'approche par niveaux se justifie aisément car les bases de données sont de plus en plus volumineuses [1] et ne tiennent plus dans la mémoire (RAM) de l'ordinateur. Son but est de minimiser le nombre d'accès à la base de données tout en offrant un temps de réponse satisfaisant. Nous avons ainsi vu l'algorithme *Apriori* qui est l'algorithme de référence pour l'extraction des motifs fréquents, l'algorithme *Pascal*, une optimisation « *presque optimale* » d'*Apriori* se basant sur le concept de classes d'équivalences et l'algorithme *Close* qui se base sur une approche clés-fermés. Nous avons découvert comment nous pouvons générer toutes les règles d'association valides à partir des motifs fréquents ainsi qu'une couverture de celles-ci. Encore une fois, ces algorithmes ne sont pas les seuls algorithmes permettant l'extraction des motifs (fermés) fréquents. Nous pouvons citer, parmi les plus efficaces, les algorithmes $CLOSET+$ [28] et $Diffset$ [29]. Cependant, ces algorithmes nécessitent l'obtention d'une représentation compacte de la base de données et donc un temps de pré-calcul non négligeable.

Pour finir, nous avons découvert deux méthodes de classification : (i) l'espace de versions et (ii) la classification bayesienne. L'espace de versions est la seule technique offrant une classification exacte mais, dans la pratique, cette méthode est peu utilisée car elle impose que la base de données ne contienne pas de données contradictoires. Quant à la classification bayesienne, c'est la plus simple à mettre en œuvre et celle qui offre le meilleur rapport « *Simplicité / Efficacité* ». Il existe d'autres méthodes de classification basées sur les règles d'association : CBA [17], $CMAR$ [16], ... Les arbres de décisions [22] sont très utilisés pour la classification sur des relations comprenant peu de tuples. Cependant ils sont très lents à construire sur des grandes bases de données : il faut parfois compter plusieurs heures pour construire un classeur qui offre une

1. source : http://www.wintercorp.com/vldb/2003_TopTen_Survey/TopTenWinners.asp

moins bonne précision que les méthodes basées sur les règles d'association.

Il est possible d'appliquer les techniques de fouille de données sur des relation binaires à des relations bases de données. Il suffit juste de reconstruire les tuples en associant chaque valeur à un identifiant qui lui est propre. Nous pouvons soit conserver la première lettre de la valeur (comme dans l'exemple ci-dessous), soit conserver la valeur elle-même si elle est unique sur toute la relation, soit préfixer la valeur par son attribut...

RowId	Sky	AirTemp	Humidity
1	Sunny	Warm	Normal
2	Sunny	Warm	Hight
3	Rain	Cold	Hight

TABLE 4.1: Relation base de données r

$\Longrightarrow$

RowId	Item
1	S W N
2	S W H
3	R C H

TABLE 4.2: Relation binaire rb

Ce sont ces techniques de transformation (ou d'autres) qui sont appliquées lorsqu'on veut extraire les motifs fréquents sur une relation base de données. Cependant, si nous changeons de problématique, alors les résultats obtenus peuvent ne pas être valides. Par exemple, nous voulons extraire tous les minimaux transversaux de r en utilisant la relation rb. D'après la définition d'un minimal transversal, le motif SR est un minimal transversal. En effet, quel que soit le tuple t de la relation rb, $t \cap SR \neq \{\emptyset\}$. Cependant, le motif SR n'est pas sémantiquement valide dans un contexte relation base de données car S et R sont deux valeurs du même attribut (Sky) et nous savons que, dans un contexte relation base de données, un attribut ne peut avoir qu'une et une seule valeur par tuple. De plus, la prise en compte de ces motifs ralentit l'exécution des algorithmes par niveaux. Ceci provient du fait que l'espace de recherche pour les problèmes de fouille de données sur des relations base de données est le treillis cube [5] et non le treillis des parties des valeurs de la relation originelle. De plus, cette structure peut aisément s'incorporer dans les SGBD-R [2] et, par conséquent, l'extraction de certaines connaissances dans les bases de données ne nécessite plus de phase de pré-traitement.

2. Enfin, ceux qui sont capables de calculer un datacube...

III

Annexes

Sommaire

5.1 Les ensembles ordonnés
5.2 Notions de Treillis
5.3 Calcul des fermés d'un treillis

Chapitre

5

Annexe

La vision du Data Mining qui vous est proposé au travers de ce manuel est basée sur les mathématiques (en existe-t-il une autre ?), un rappel sur les divers concepts mathématiques que nous manipulerons par la suite peut s'avérer nécessaire pour le lecteur. Après avoir introduit les ensembles ordonnés, nous verrons comment les représenter, puis définirons leurs bornes. Nous verrons quels sont les ordres utiles pour le Data Mining et parlerons de contraintes et d'espaces convexes. Nous aborderons ensuite la notion de treillis et de fermeture et verrons deux algorithmes pour le calcul des fermés.

5.1 Les ensembles ordonnés

5.1.1 Qu'est ce qu'un ensemble ordonné ?

Un **ensemble ordonné** (ou *ordre partiel*) est un couple $\mathcal{P} = \langle E, \leq_{\mathcal{P}} \rangle$ où E est un ensemble d'éléments et $\leq_{\mathcal{P}}$ est un ordre sur E qui vérifie $\forall a, b, c \in \mathcal{P}$:

- l'antisymétrie : si $a \leq_{\mathcal{P}} b$ alors on n'a pas $b \leq_{\mathcal{P}} a$;
- la reflexivité : on a $a \leq_{\mathcal{P}} a$;
- la transitivité : si $a \leq_{\mathcal{P}} b$ et $b \leq_{\mathcal{P}} c$, alors $a \leq_{\mathcal{P}} c$.

Convention : Soit $e \in E$, alors nous noterons aussi $e \in \mathcal{P}$.

Exemple 5.1 L'ensemble $E = \{\{A\}, \{A, B\}, \{C\}, \{A, B, C\}\}$ peut être ordonné en utilisant l'ordre d'inclusion.

Définition 5.1 Relation de couverture
Soit $a, b, c \in \mathcal{P}$, alors nous dirons que a couvre b (ou b est couvert par a), noté $b \lessdot a$, si et seulement si $b \leq a$ et $\nexists c$ tel que $b < c < a$.

Exemple 5.2 En utilisant l'ensemble ordonné précédemment introduit, nous avons $\{A\} \lessdot \{A, B\}$, par contre $\{A\} \not\lessdot \{A, B, C\}$ car $\{A\} \subset \{A, B\} \subset \{A, B, C\}$.

Définition 5.2 Éléments comparables
Soit $a, b \in \mathcal{P}$, a et b sont dits comparables (noté $a \sim b$) si et seulement si $a \leq b$ ou $b \leq a$. Réciproquement a et b sont dits incomparables (noté $a||b$) si et seulement si $a \not\leq b$ et $b \not\leq a$.

Exemple 5.3 En utilisant l'ensemble ordonné précédemment introduit, nous avons $\{A\} \sim \{A, B\}$ et $\{A\}||\{C\}$.

Définition 5.3 Chaîne et Anti-chaîne

1. Soit $\mathcal{C} \subseteq E$. $\mathcal{C}$ est une chaîne de $\mathcal{P}$ si et seulement si $\forall\ a, b \in C$ nous avons : $a \sim b$. $\mathcal{C}$ est une chaîne maximale de $\mathcal{P}$ si et seulement si (i) $\mathcal{C}$ est une chaîne et (ii) il n'existe aucune chaîne de $\mathcal{P}$ qui contienne $\mathcal{C}$. La hauteur de $\mathcal{P}$ (notée $haut(\mathcal{P})$) est la taille de la plus longue chaîne de $\mathcal{P}$.

2. Soit $\mathcal{A} \subseteq E$. $\mathcal{A}$ est une **anti-chaîne** si et seulement si $\forall a, b \in \mathcal{A}$ nous avons $a||b$. $\mathcal{A}$ est une **anti-chaîne maximale** de $\mathcal{P}$ si et seulement si $\mathcal{A}$ est une anti-chaîne de $\mathcal{P}$ et $\nexists \mathcal{A}'$ anti-chaîne de $\mathcal{P}$ telle que $\mathcal{A} \subset \mathcal{A}'$. La **largeur de** $\mathcal{P}$ (notée $larg(\mathcal{P})$ est la taille de la plus longue anti-chaîne de $\mathcal{P}$.

Remarque : Les chaînes maximales et anti-chaînes maximales ne sont pas forcément uniques pour un ensemble ordonné quelconque!

Exemple 5.4 En utilisant l'ensemble ordonné précédemment introduit, les chaînes $\{\{A\}, \{A, B\}, \{A, B, C\}\}$ et $\{\{C\}, \{A, B, C\}\}$ sont deux chaînes maximales et $\{\{A\}$,
$\{C\}\}$ est une anti-chaîne.

5.1.2 Représentation graphique

À tout ensemble ordonné $\mathcal{P} = \langle E, \leq \rangle$, nous pouvons associer deux graphes orientés :

1. le graphe $G_f(\mathcal{P} = (E, E_f))$ dont les sommets sont les éléments de $\leq_{\mathcal{P}}$. Il est appelé **fermeture transitive de** $\mathcal{P}$

Exemple 5.5
Soit $E = \{a, b, c, d, e, f, g\}$ et $\leq$ un ordre sur E. nous pouvons définir l'ensemble ordonné $\mathcal{P}$ en utilisant les règles suivantes :

- $a||b, c||d||e, f||g$
- $a \lessdot c, a \lessdot d, b \lessdot d, b \lessdot e, c \lessdot f, d \lessdot f, d \lessdot g, e \lessdot g$
- $a \leq f, b \leq g$.

Alors la figure ci-dessous représente la fermeture transitive de $\mathcal{P}$

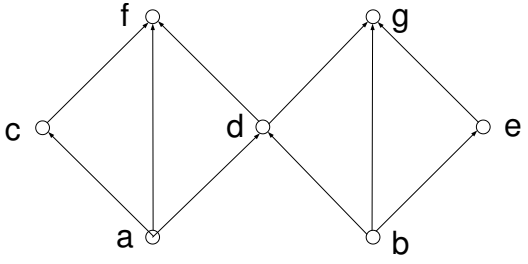


FIGURE 5.1: Fermeture transitive de $\mathcal{P}$

Exercice 5.1
Dans la définition de l'ensemble ordonné précédent, est-il nécessaire d'avoir : « $a \leq f, b \leq g$ » ?

2. Le graphe $G_h(\mathcal{P}) = (E, E_h)$, appelé **graphe de couverture** (ou *diagramme de Hasse*), ayant pour sommets les éléments de E et pour arcs les éléments de $\lessdot$. Ce graphe comporte un nombre minimum d'arcs.

Exemple 5.6
avec l'exemple précédent, nous obtenons le graphe suivant :

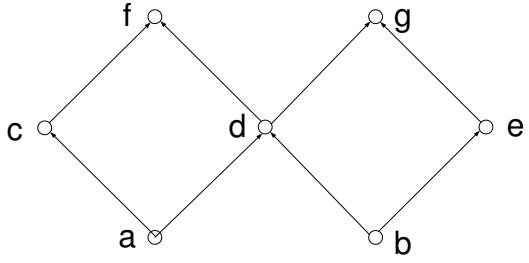


FIGURE 5.2: Graphe de couverture de $\mathcal{P}$

Convention : Dans la suite de ce manuel, chacun des ensembles ordonnés sera représenté par son diagramme de Hasse afin d'éviter de surcharger les figures.

5.1.3 Bornes dans un ordre

On définit les ensembles suivants relatifs à un élément a de E :

1. L'ensemble des **successeurs** : $succ_{\mathcal{P}}(a) = \{b \in E \mid a <_{\mathcal{P}} b\}$,

2. l'ensemble des **successeurs directs** : $imsucc_{\mathcal{P}}(a) = \{b \in E \mid a \lessdot_{\mathcal{P}} b\}$.

Dualement, nous notons respectivement $pred_{\mathcal{P}}(a)$ et $impred_{\mathcal{P}}(a)$ les **prédécesseurs** et **prédécesseurs directs** de a.

Soit X un sous-ensemble de $\mathcal{P} = \langle E, \leq_{\mathcal{P}} \rangle$ tel que $|X| \geq 2$ (X contient au moins deux éléments).

- Un élément a de X est dit **minimal** (pour la relation d'ordre $\leq_{\mathcal{P}}$) s'il n'existe pas $b \in X$ tel que $b \leq a$. L'ensemble des minimaux de X est noté $min(X)$.
- Un élément a de X est dit **maximal** (pour la relation d'ordre $\leq_{\mathcal{P}}$) s'il n'existe pas $b \in X$ tel que $a \leq b$. L'ensemble des maximaux de X est noté $max(X)$.

Exercice 5.2
En considérant l'ensemble ordonné défini dans l'exercice 5.5, que contiennent les ensembles $max(E)$ et $min(E)$?

Définition 5.4 Minorant, Majorant
L'ensemble des minorants de X est défini par $l(X) = \{a \in \mathcal{P} \mid a \leq b, \forall b \in X\}$.
L'ensemble des majorants de X est défini par $u(X) = \{a \in \mathcal{P} \mid b \leq a, \forall b \in X\}$.

Exemple 5.7 En considérant l'ensemble ordonné défini dans l'exercice 5.5, alors les ensembles $u(E)$ et $l(E)$ ne contiennent aucun élément. Par contre, soit $X = \{a, c, d, f\}$, alors $u(X) = \{f\}$ et $l(X) = \{a\}$.

Définition 5.5 Borne inférieure, Borne Supérieure
Si $l(X)$ admet un plus grand élément dans $\mathcal{P}$, il est unique et appelé borne inférieure de X. De même, si $u(X)$ admet un plus petit élément dans $\mathcal{P}$, il est unique et appelé borne supérieure de X.

Notations : La borne inférieure de X est notée $inf(X)$ ou $\bigwedge X$, la borne supérieure est notée $sup(X)$ ou $\bigvee X$. Si $X = \{a, b\}$, nous noterons $a \bigwedge b$ et $a \bigvee b$.

La propriété suivante exprime le fait qu'un ordre peut être entièrement défini par l'un des opérateurs $\bigwedge$ ou $\bigvee$.

Propriété 5.1 [27]
Étant donné un ensemble ordonné $\mathcal{P} = \langle E, \leq_{\mathcal{P}} \rangle$ on a : $a \leq b \Leftrightarrow a = a \bigwedge b \Leftrightarrow b = a \bigvee b$.

5.1.4 Ordres utiles pour le data mining

Parmi les divers ordres existant en mathématiques, quatre se révèlent particulièrement très utiles pour le data mining : l'inclusion, l'ordre lexicographique, l'ordre lectique et l'ordre de généralisation. Seuls les trois premiers seront étudiés dans ce manuel. En ce qui concerne l'ordre de généralisation, le lecteur pourra se référer à [21, 6].

Ordre d'inclusion

Soit X et Y deux ensembles. On dit que X est inclus dans Y (noté $X \subseteq Y$) si tout élément de X est aussi un élément de Y. On dit aussi que X est contenu dans Y, ou que X est un sous-ensemble de Y, ou que X est une partie de Y.

Exemple 5.8 La figure ci-dessous représente le graphe de couverture de tous les sous-ensembles de $\{A, B, C, D\}$ ordonnés selon l'inclusion.

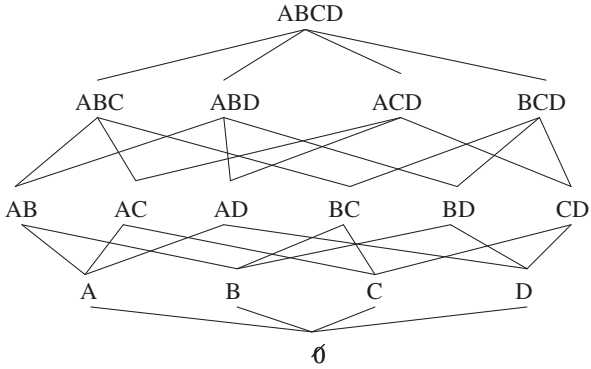


FIGURE 5.3: Diagramme de Hasse des sous-ensembles de $\{A, B, C, D\}$ selon $\subseteq$.

Ordre lexicographique

L'ordre lexicographique, noté $<_{lex}$, est l'ordre du dictionnaire. Contrairement à l'inclusion, tous les éléments peuvent être totalement ordonné avec cet ordre (i.e. il n'existe pas d'éléments incomparables), ce qui ne devrait surprendre personne!

Exemple 5.9 Soit $E = \{A, B, C\}$, alors tous les sous-ensembles de E sont ordonnés de la façon suivante en utilisant l'ordre lexicographique : $\{\emptyset\} <_{lex} \{A\} <_{lex} \{A, B\} <_{lex} \{A, B, C\} <_{lex} \{A, C\} <_{lex} \{B\} <_{lex} \{B, C\} <_{lex} \{C\}$

Cependant, il est possible de donner une définition « *plus mathématique* » de cet ordre :

Définition 5.6 Ordre lexicographique
Soit E un ensemble d'éléments. Supposons que les éléments sont totalement ordonnés et donc comparables deux à deux via un ordre noté $\preceq$. Soit X et $Y \subseteq E$, alors nous avons : $X <_{lex} Y \Leftrightarrow min_{\preceq}(X \backslash (X \cap Y)) \preceq min_{\preceq}(Y \backslash (X \cap Y))$. Autrement dit, on prive X et Y de leur partie commune et on regarde si le premier élément de X est plus petit (selon $\preceq$) que le premier élément de Y.

De plus, comme le montre la figure 5.4, l'ordre lexicographique est représentable par un arbre. Le sens de lecture, dans cet arbre, est de gauche à droite et de bas en haut. Ainsi, tous les noeuds (éléments) des sous-arbres relatifs à un noeud X (élément) sont des successeurs de X selon l'ordre lex.

Ordre lectique

Moins connu par le grand public que l'ordre lexicographique, l'ordre lectique (ou ordre lexicographique inverse), noté $<_{lec}$, est un ordre important pour le data mining. Comme nous pouvons le voir sur la figure suivante, la représentation graphique de cet ordre est l'image miroir de l'ordre précédent.

Comme nous pouvons l'apercevoir, les représentations graphiques des ordres lectiques et lexicographiques sont «*assez proches* ». C'est pourquoi, nous ne nous étonnerons pas que leur définition le soient aussi. En fait, il suffit de changer l'opérateur min par l'opérateur max dans la définition 5.6 pour obtenir celle de l'ordre lectique.

Définition 5.7 Ordre lectique
Soit E un ensemble d'éléments. Supposons que les éléments sont totalement

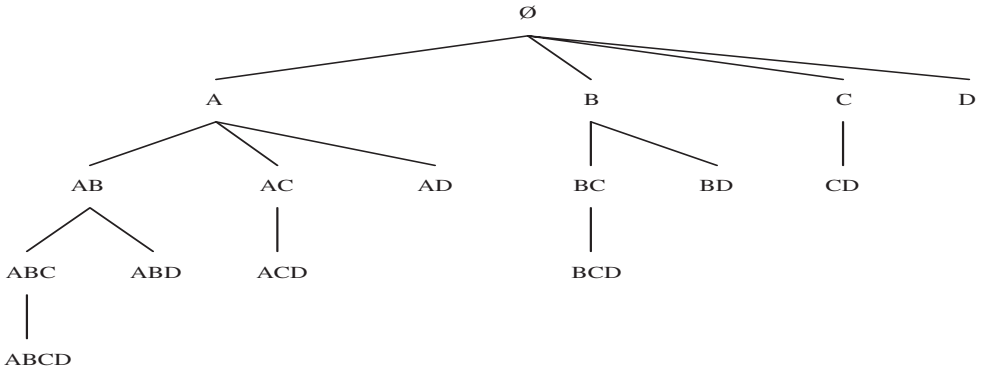


FIGURE 5.4: Représentation par arbre des sous-ensembles de $\{A, B, C, D\}$ selon l'ordre *lex*.

ordonnés et donc comparables deux à deux via un ordre noté $\preceq$. Soit X et $Y \subseteq E$, alors nous avons : $X <_{lec} Y \Leftrightarrow max_{\preceq}(X \backslash (X \cap Y)) \preceq max_{\preceq}(Y \backslash (X \cap Y))$. Autrement dit, on prive X et Y de leur partie commune et on regarde si le dernier élément de X est plus petit (selon l'ordre $\preceq$) que le dernier élément de Y.

Encore une fois, tous les sous-ensembles d'un ensemble E peuvent être totalement ordonnés en utilisant l'ordre lectique comme le montre l'exemple suivant.

Exemple 5.10 Soit $E = \{A, B, C\}$, alors tous les sous-ensembles de E sont ordonnés de la façon suivante en utilisant l'ordre lectique : $\{\emptyset\} <_{lec} \{A\} <_{lec} \{B\} <_{lec} \{A, B\} <_{lec} \{C\} <_{lec} \{A, C\} <_{lec} \{B, C\} <_{lec} \{A, B, C\}$

Exercice 5.3
En considérant l'ensemble ordonné de l'exercice 5.1, quels sont les éléments minimaux et maximaux selon chacune des trois relations d'ordre.

5.1.5 Contraintes et Espaces convexes

Lorsque nous voulons extraire des connaissances à partir d'une base de données, nous connaissons *a-priori* quel(s) type(s) de connaissances nous voulons

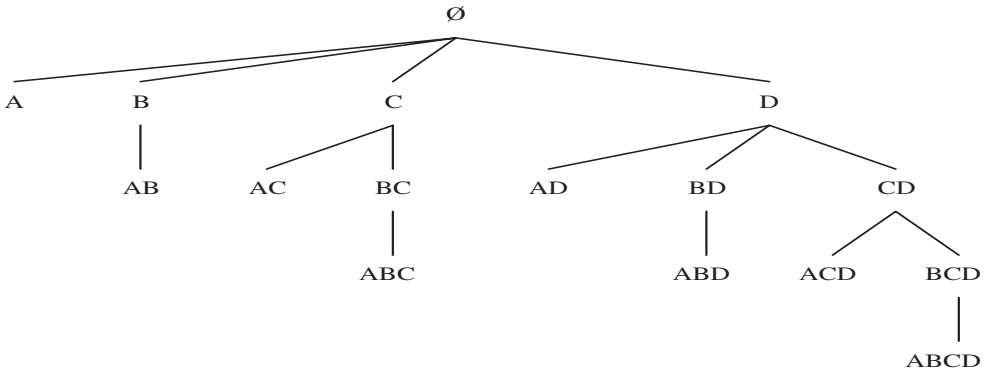


FIGURE 5.5: Représentation par arbre des sous-ensembles de $\{A, B, C, D\}$ selon l'ordre *lec*.

extraire. Par exemple, nous pouvons extraire tous les sous-ensembles des valeurs d'une relation qui apparaissent au moins 4 fois dans cette relation. La phrase « *extraire tous les sous ensembles des valeurs d'une relation qui apparaissent au moins 4 fois dans cette relation* » est appelée **contrainte**. Parmi les divers types de contraintes qui existent, deux nous serons utiles :

Définition 5.8 Contrainte monotone/antimonotone
Soit E un ensemble et $X \subseteq E$, une contrainte *cont* est dite monotone selon l'ordre $\preceq$ si et seulement si : X satisfait *cont*, alors tous les successeurs de X satisfont aussi *cont* ($cont(X) \Rightarrow cont(Y), \forall X \preceq Y$). De même, une contrainte *cont* est dite antimonotone selon l'ordre inclusion si et seulement si : X satisfait *cont*, alors tous les prédécesseurs de X satisfont aussi *cont* ($cont(X) \Rightarrow cont(Y), \forall Y \preceq X$).

Exemple 5.11 La contrainte « *extraire tous les sous ensembles des valeurs d'une relation qui apparaissent au moins 4 fois dans cette relation* » est une contrainte anti-monotone. En effet, par exemple, si un motif $\{A, B, C, D\}$ apparaît au moins 4 fois, alors le motif $\{A, B, C\}$ apparaît au moins 4 fois !! Cette idée sera reprise dans le chapitre **??**.

Heureusement, si nous avons réussi à identifier la contrainte sous-jacente à

notre problème, alors nous n'avons pas besoin d'extraire tous les motifs solutions. Seuls les éléments minimaux et maximaux selon l'ordre d'inclusion nous sont nécessaire car l'ensemble des motifs solutions est alors un espace convexe.

Définition 5.9 Espace convexe
Soit $\langle E, \preceq \rangle$ un ensemble partiellement ordonné, $C \subseteq E$ est un espace convexe si $\forall x, y, z \in P, x \preceq y \preceq z$ et $x, z \in C \Rightarrow y \in C$. Donc C est borné par deux ensembles : un majorant ou "upper set" défini par $max_{\preceq}(C)$ et un minorant ou "lower set" défini par $min_{\preceq}(C)$.

Propriété 5.2 Soit *cont* une contrainte, E un ensemble :

- si *cont* est monotone alors seul le minorant nous intéresse, le majorant étant $\{E\}$ lui même.
- si *cont* est anti-monotone alors seul le majorant nous intéresse, le minorant étant $\{\emptyset\}$.

5.2 Notions de Treillis

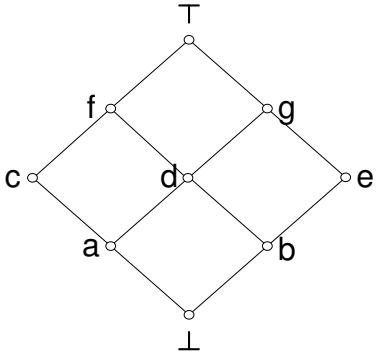
Définition 5.10 Treillis
Soit $\mathcal{L} = \langle E, \preceq \rangle$ un ordre non vide. $\mathcal{L}$ est appelé **treillis** (en anglais **lattice**) si pour tout couple $(x, y) \in (\mathcal{L} \times \mathcal{L})$ les bornes supérieures et inférieures de x et y existent. $\bigwedge E$ est appelé **top** et noté $\top$, $\bigvee E$ est appelé **bottom** et noté $\bot$.

Convention : On représentera un treillis selon la convention des ordres : en plaçant l'élément $\bot$ en bas et $\top$ en haut, induisant une orientation implicite permettant de supprimer les flèches du diagramme de Hasse : soit a,b $\in \mathcal{L}$, si $a \preceq b$ alors a sera plus bas que b dans le diagramme de Hasse de $\mathcal{L}$.

Cette définition est *relationnelle* (ou *« combinatoire »*) car elle fait intervenir les propriétés de la relation d'ordre $\leq_{\mathcal{L}}$. Il existe une autre définition d'un treillis dite *algébrique* parce qu'elle fait intervenir les propriétés des opérateurs $\bigwedge$ et $\bigvee$ définis sur l'ensemble $\mathcal{L}$.

Définition 5.11 Treillis (V2)
Un treillis $\langle E, \preceq \rangle$ est un ensemble $\mathcal{L}$ muni de deux opérateurs notées $\bigvee$ et $\bigwedge$ vérifiant les axiomes suivants pour tout couple d'éléments $(x, y) \in (\mathcal{L} \times \mathcal{L})$:

idempotence	A1 : $x \bigvee x = x$ A1' : $x \bigwedge x = x$
commutativité	A2 : $x \bigvee y = y \bigvee x$ A2' : $x \bigwedge y = y \bigwedge x$
associativité	A3 : $(x \bigvee y) \bigvee z = x \bigvee (y \bigvee z)$ A3' : $(x \bigwedge y) \bigwedge z = x \bigwedge (y \bigwedge z)$
absorption	A4 : $x \bigwedge (x \bigvee y) = x$ A4' : $x \bigvee (x \bigwedge y) = x$



Exemple 5.12

FIGURE 5.6: Un trellis $\mathcal{L}$

Remarque : La représentation de ce treillis est presque similaire à celle de l'ensemble ordonné de la figure 5.2 (p. 78), nous pouvons voir l'apparition d'un élément top et d'un élément bottom. L'absence de ces deux éléments a pour conséquence : $u(E) = l(E) = \{\emptyset\}$ (cf. exemple 5.7).

5.2.1 Principe de dualité et treillis

Soit $\langle E, \leq \rangle$ un treillis munis des opérateurs $\bigvee$ et $\bigwedge$; considérons l'ensemble $\mathcal{L}$ muni de l'ordre dual $\geq$. Par dualité, $\langle E, \geq \rangle$ est aussi un treillis. Notons $\sqcap$ et $\sqcup$ les opérateurs de bornes supérieures et inférieures de ce treillis. Alors, $\forall x, y \in E : x \sqcap y = x \bigvee y$ et $x \sqcup y = x \bigwedge y$. Ceci résulte de la définition 5.10

d'un treillis.

Supposons que dans un treillis quelconque, nous ayons démontré (à partir des seuls axiomes d'un treillis) une propriété (P) faisant intervenir des opérations $\bigvee$ et $\bigwedge$ de la relation $\leq$, alors cette propriété restera vérifiée dans le treillis dual. Dans ce treillis elle s'exprime en changeant dans (P) les symboles $\bigvee$ par $\bigwedge$, $\bigwedge$ par $\bigvee$ et en remplaçant les relations d'ordre $\leq$ par $\geq$. On dit qu'on a ainsi obtenu la propriété (P') duale de (P). Le **principe de dualité** s'énonce alors :
La duale d'une propriété vraie pour tout treillis est également une propriété vraie pour tout treillis

Exemple 5.13 Les axiomes A_i et A_i' servant à définir un treillis sont duaux.

Le lecteur pourra se référer à [2, 11] pour plus de précisions.

5.2.2 Éléments irréductibles dans un trellis

Tout élément x d'un treillis peut s'écrire sous la forme $x = y \bigwedge z$ (cf. propriété 5.1). Prenons par exemple $y = x$ et $z \geq x$. Cependant, une telle « *décomposition* » de x ne fournit aucune information sur x; seules sont intéressantes les représentations de la forme $x = y \bigwedge z, y; z > x$.

Définition 5.12 Élément inf-irréductible
En conséquence , un élément x d'un treillis $\mathcal{L} = \langle E, \leq \rangle$ est dit inf-irréductible, aussi noté $\bigwedge$-irréductible, si :

i $x \neq \top$

ii $\forall y, z \in \mathcal{L}, x = y \bigwedge z \Rightarrow x = y$ ou $x = z \forall y, z \in \mathcal{L}$

ii' $y > x$ et $z > x \Rightarrow y \bigwedge z > x, \forall y, z \in \mathcal{L}$

Remarque : Les définitions ii et ii' sont équivalentes.

Conséquence : Les inf-irréductibles ne sont couverts que par un seul élément. Donc, dans le diagramme de Hasse représentant le treillis, un inf-irréductible n'aura qu'un seul arc sortant.

Définition 5.13 Élément sup-irréductible
Dualement, un élément x d'un treillis $\mathcal{L} = \langle E, \leq \rangle$ est dit sup-irréductible, aussi noté $\bigvee$-irréductible, si :

i $x \neq \bot$

ii $\forall y, z \in X, x = y \bigvee z \Rightarrow x = y$ ou $x = z, \forall y, z \in \mathcal{L}$

ii' $y < xetz < x \Rightarrow y \bigvee z < x, \forall y, z \in \mathcal{L}$

Remarque : Comme précédemment, les définitions ii et ii' sont équivalentes. Nous pouvons aussi remarquer que les définitions des inf-irréductibles et des sup irréductibles sont duales. Par conséquent, les sup-irréductibles ne couvrent qu'un seul élément et, dans le diagramme de Hasse représentant le treillis, un sup-irréductible n'aura qu'un seul arc entrant.

Notation : En anglais les éléments sup-irréductibles sont appelés *join-irreducible* et les inf-irréductibles sont quant à eux appelés *meet-irreducible*. On note, pour un treillis $\mathcal{L} = \langle E, \leq \rangle$, $J(\mathcal{L})$ l'ensemble des sup-irréductibles et $M(\mathcal{L})$ celui des inf-irréductibles.

Exemple 5.14 Sur le treillis précédent, les sup-irréductibles sont $\{a, b, c, e\}$ (représentés en noir sur la figure 5.7) et les inf-irréductibles sont $\{c, e, f, g\}$ (représentés en noir sur la figure 5.8)

Remarque : c et e sont en même temps inf-irréductibles et sup-irréductibles.

Le théorème ci-dessous indique qu'il est possible de retrouver tous les éléments du treillis à partir des irréductibles (inf-irréductibles et/ou sup-irréductibles).

Theorème 5.1
Soit $\mathcal{L}$ un treillis, alors nous avons :

1. $\forall a \in \mathcal{L}, a = \bigvee \{x \subseteq \mathcal{J}(\mathcal{L}) | x \leq a\}$
2. $\forall a \in \mathcal{L}, a = \bigwedge \{a \subseteq \mathcal{M}(\mathcal{L}) | a \leq x\}$

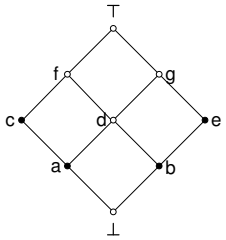


FIGURE 5.7: sup-irréductibles

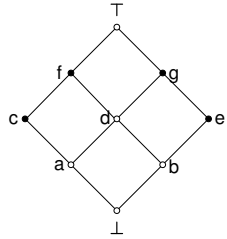


FIGURE 5.8: inf-irréductibles

Remarque : Par conséquent, comme il sera rappelé dans la suite de cet ouvrage, la connaissance des éléments irréductibles, et particulièrement des inf-irréductibles, est très importante !!!

Exemple 5.15 En reprenant l'exemple de la figure 5.8, on peut retrouver l'élément $d = \bigwedge\{f, g\} = \bigvee\{a, b\}$ et l'élément $\bot = \bigwedge\{c, e, f, g\}$.

5.2.3 Fermeture et treillis

Le but de cet ouvrage n'étant pas une étude approfondie de l'algorithmique des treillis, mais plus une utilisation judicieuse de ceux-ci pour le Data Mining, nous n'étudierons que les treillis ayant l'inclusion ($\subseteq$) comme relation d'ordre.

Définition 5.14 Opérateur de fermeture, élément fermé, système de fermeture

- Soit E un ensemble fini, alors $\mathcal{P}(E)$ désigne l'ensemble des parties de E. Autrement dit, $\mathcal{P}(E)$ contient tous les sous-ensembles de E, y compris E lui même. Une application $h : \mathcal{P}(E) \to \mathcal{P}(E)$ est un **opérateur de fermeture** (ou une **fermeture**) sur E, si $\forall a, b \subseteq E$:

  1. $a \subseteq h(a)$ (**extensivité**)
  2. si $a \subseteq b$, alors $h(a) \subseteq h(b)$ (**isotonie**)
  3. $h \circ h(a) = h(a)$ (**idempotence**)

- Soit $a \in \mathcal{P}(E)$, si $a = h(a)$, alors a est appelé un **fermé** .
- L'ensemble $\{a \subseteq E \mid a = h(a)\}$ est appelé le **système de fermeture** de h pour E. Donc un système de fermeture ne contient que des fermés.

Exemple 5.16 L'exemple le plus simple d'un opérateur de fermeture est l'application identité : $Id : \mathcal{P}(E) \to \mathcal{P}(E) : a \mapsto a$. L'ensemble des fermés pour cet opérateur est l'ensemble des $2^{|E|}$ sous-ensembles de E. Ainsi, la figure 5.3 est un treillis plus communément appelé treillis des parties de l'ensemble $\{A, B, C, D\}$.

Theorème 5.2
[4] Soit $\mathcal{L} = \langle E, \preceq \rangle$ un treillis et h un opérateur de fermeture quelconque sur $\mathcal{L}$. L'ensemble partiellement ordonné $\mathcal{F}_h = \langle \{x \in \mathcal{L} \mid h(x) = x\}, \preceq \rangle$ est un treillis et nous avons pour tout $A \subseteq \mathcal{F}_h$:

$$\bigwedge_{\mathcal{F}_h} A = \bigwedge_{\mathcal{L}} A$$

$$\bigvee_{\mathcal{F}_h} A = h(\bigvee_{\mathcal{L}} A)$$

Ce dernier théorème montre qu'à chaque opérateur de fermeture, nous pouvons associer un treillis appelé **treillis des fermés**.

5.2.4 Treillis de Galois (des fermés)

Soit deux ensembles $\mathcal{O}$ et $\mathcal{A}$, $\mathcal{O}$ représente un ensembles d'objets (plus communément appelé **tuples**) et $\mathcal{A}$ un ensemble d'**attributs** (ou d'items). Un **contexte** est un triplet $r = (\mathcal{A}, \rho, \mathcal{O})$ où ρ est un sous-ensemble de $\mathcal{A} \times \mathcal{O}$.

Exemple 5.17 Un exemple de contexte $\mathcal{R} = (\mathcal{A}, \rho, \mathcal{O})$, $\mathcal{A} = \{A, B, C, D, E\}$ et $\mathcal{O} = \{1, 2, 3, 4\}$, est donné par la représentation matricielle [1] suivante :

$\mathcal{A} \times \mathcal{O}$	A	B	C	D	E
1	1	0	1	1	0
2	1	1	1	0	1
3	0	1	1	0	1
4	0	1	0	0	1

1. Les 1 représentent les couples (a, o) tel que $a \rho o$.

Cette représentation matricielle est appelée **relation binaire** du contexte r. Une autre representation matricielle de de contexte est :

RowId	Item
1	ACD
2	ABCE
3	BCE
4	BE

Notation : (*i*) Cette dernière représentation étant plus claire que la première, nous l'utiliserons dans la suite de ce document.
(*ii*) De plus, nous noterons les éléments de $\mathcal{A}$ par des lettres, et ceux de $\mathcal{O}$ par des chiffres. Ceci provient du fait les chercheurs en « *base de donnés* » ont l'habitude de numéroter les lignes (aussi appelés tuples) par des chiffres et les attributs (ou items) par des lettres.

Définition 5.15 Opérateur de fermeture sur une relation binaire
Soit $X \subseteq \mathcal{A}$ et r une relation binaire sur $\mathcal{A}$. Alors nous pouvons définir un opérateur de fermeture h sur r de la manière suivante : $h(X) = \cap_i t_i \mid X \subseteq t_i$. Par conséquent, pour calculer la fermeture d'un ensemble X, il faut faire l'intersection de tous les tuples de la relation binaire contenant X. Si $X = h(X)$ alors par analogie avec la définition 5.14 X est un **fermé**.

Exemple 5.18 Avec la relation binaire de l'exemple 5.17, nous avons :
$h(\{A\}) = \{A, C, D\} \cap \{A, B, C, E\} = \{A, C\}$.

Exercice 5.4
Toujours en considérant la relation binaire de l'exemple 5.17, calculer les autres fermetures de chacun des sous-ensembles de $\{A, B, C, D, E\}$. Que constatez vous ?

Définition 5.16 Clé
Soit $X \subseteq \mathcal{A}$ un fermé, $Y \subseteq X$ et h un opérateur de fermeture. Y est une clé pour X si et seulement si $Y \in min_{\subseteq}\{Z \subseteq X : h(Z) = X\}$. Notons qu'un fermé peut avoir plusieurs clés.

Exemple 5.19 Avec la relation binaire de l'exemple 5.17, puisque nous avons : $h(\{A\}) = \{A, C\}$ et $h(\{\emptyset\}) = \{\emptyset\}$, alors $\{A\}$ est une clé pour $\{A, C\}$.

Exercice 5.5
En considérant le fermé $\{A, C\}$, peut on dire que $\{C\}$ est aussi une clé pour ce fermé ?

Propriété 5.3 Soit $X \subseteq \mathcal{A}$ et h un opérateur de fermeture sur $\mathcal{A}$. La contrainte « *X est une clé* » est une contrainte antimonotone selon l'ordre d'inclusion. Autrement dit tout sous-ensemble d'une clé est aussi une clé (mais pour un autre fermé).

Remarque : Il existe une relation entre les clés d'un fermé et les clés primaires d'une base de donnés. Cette relation sera développée dans le chapitre suivant.

Notation : Dans la suite de ce document, les accolades servant à définir des ensembles ne seront présentes que lorsqu'il sera nécessaire de les préciser. Avec l'exemple précédent, nous avons $h(A) = AC$.

Puisque nous avons un opérateur de fermeture h sur une relation binaire r, nous pouvons pouvons appliquer le théorème de Birkhoff (cf. théorème 5.2) afin de construire le treillis de Galois.

Définition 5.17 Treillis de Galois (des fermés)
Soit $X \subseteq \mathcal{A}$ et r une relation binaire sur $\mathcal{A}$. L'ensemble $h(r) = \{X \mid X = h(X)\}$ ordonné par la relation d'inclusion ($\subseteq$) est un treillis appelé treillis de Galois ou treillis des fermés. De plus, dans ce treillis, les opérateurs Join ($\bigvee$) et Meet ($\bigwedge$) s'expriment de la manière suivante :

$$\forall X_1, .., X_n \subseteq \mathcal{P}(\mathcal{A}), \bigvee_i X_i = h(\bigcup_i X_i) \text{ et } \bigwedge_i X_i = \bigcap_i X_i.$$

Exemple 5.20 Avec la relation binaire de l'exemple 5.17, nous obtenons le diagramme suivant :

De plus, $AC \bigvee BE = h(AC \cup BE) = h(ABCE) = ABCE$ et $ACD \bigwedge ABCE = ACD \cap ABCE = AC$.

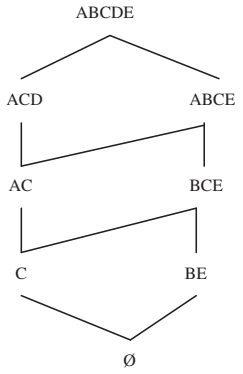


FIGURE 5.9: Treillis de Galois de la relation r

5.3 Calcul des fermés d'un treillis

Comme nous le verrons dans la suite du document, la génération des fermés est primordiale en Data Mining. Nous allons voir deux algorithmes pour le calcul des fermés : l'algorithme de Norris [23] (cf. algorithme 15, p. 93) et l'algorithme Next-Closure [11] (cf. algorithme 16, p. 95). L'algorithme de Norris a l'avantage d'être assez simple, mais ne peut calculer les fermés que pour un système de fermeture stable par intersection. Par contre, l'algorithme Next-Closure, peut être appliqué à n'importe quel système de fermeture. Nous l'appliquerons dans ce chapitre pour un système de fermeture stable par intersection (e.g. une relation binaire) et nous verrons une illustration de cet algorithme dans le chapitre concernant les dépendances fonctionnelles (cf chapitre **??**). De plus, dans ce dernier algorithme, chaque fermé est généré une et une seule fois.

5.3.1 Méthode de Norris

D'après la definition 5.17, chaque élément du treillis de Galois peut être obtenu par une suite d'intersection des tuples de la relation. C'est pourquoi nous disons que le treillis de Galois est un **système de fermeture par intersection**. L'idée de la méthode de Norris est de calculer, via une suite d'intersections, l'ensemble des fermés. Au début de l'algorithme, la suite $\mathcal{S}$ est initialisée avec l'élément $\top$, puis nous faisons une intersection entre $\mathcal{S}$ et le premier tuple t_1

et ajoutons le résultat à $\mathcal{S}$. Ensuite nous répétons ce processus jusqu'à ce que tous les tuples aient été examinés.

Alg. 15 Méthode de Norris

Entrée : ensemble d'items $\mathcal{A}$, une relation binaire r
Sortie : S ensemble des fermés
1: $\mathcal{S} = \mathcal{A}$
2: **pour tout** $t_i \in r$ **faire**
3: $\quad \mathcal{S} = (\mathcal{S} \cap t_i) \cup \mathcal{S}$
4: **fin pour**
5: **return** $\mathcal{S}$

Exemple 5.21 Avec la relation binaire de l'exemple 5.17, voici une trace de l'algorithme précédent :

1. $\mathcal{S} = \{ABCDE\}$
2. $\mathcal{S} = (\{ABCDE\} \cap ACD) \cup \{ABCDE\} = \{ACD, ABCDE\}$
3. $\mathcal{S} = (\{ACD, ABCDE\} \cap ABCE) \cup \{ACD, ABCDE\} = \{AC, ABCE, ACD, ABCDE\}$
4. $\mathcal{S} = (\{AC, ABCE, ACD, ABCDE\} \cap BCE) \cup \{AC, ABCE, ACD, ABCDE\} = \{C, BCE, AC, ABCE, ACD, ABCDE\}$

   Remarque : l'élément C est obtenu de deux façons différentes : soit avec $AC \cap BCE$ soit avec $ACD \cap BCE$, mais pourtant n'est présent qu'une seule fois dans $\mathcal{S}$.
5. $\mathcal{S} = (\{C, BCE, AC, ABCE, ACD, ABCDE\} \cap BE) \cup \{C, BCE, AC, ABCE, ACD, ABCDE\} = \{\emptyset, BE, C, BCE, AC, ABCE, ACD, ABCDE\}$

Exercice 5.6
En considérant la relation binaire suivante, utiliser la méthode précédente pour trouver l'ensemble des fermés, puis construisez le treillis de Galois associé.

RowId	Item
1	BC
2	AB
3	AC
4	BCD

5.3.2 Algorithme Next-Closure

Comme nous l'avons vu dans le paragraphe précédent, l'algorithme de Norris (cf. algorithme 15) peut générer plusieurs fois le même fermé, c'est pourquoi nous allons introduire l'algorithme Next-Closure qui ne calcule chaque fermé qu'une et une seule fois. De plus, à la sortie de l'algorithme, l'ensemble des fermés est ordonné selon l'ordre lectique (cf. définition 5.7).

Définition 5.18 Fonction Inc
Étant donné un ensemble (d'attributs) $\mathcal{A} = \{a, b, c...\}$ et $X \subseteq \mathcal{A}$, alors $Inc(X, y) = X \cup y \backslash \{a, ..., x\}$. Bien sûr $y \in \mathcal{A} \backslash X$ et x désigne le prédécesseur immédiat de y selon l'ordre lexicographique.

Exemple 5.22 Soit $\mathcal{A} = \{A, B, C, D, E\}, X = \{A, C, D\}$ alors nous avons :

- $Inc(X, B) = \{C, D, E\}$
- $Inc(X, E) = \{E\}$

Remarque : Le successeur immédiat de X (selon l'ordre lectique) est obtenu avec $x = min_{<_{lex}} \mathcal{A} \backslash X$.

L'algorithme de calcul des fermés Next-Closure est basé sur le théorème suivant [11] :

Theorème 5.3
Soit X un sous-ensemble de $\mathcal{A}$ et h un opérateur de fermeture sur $\mathcal{A}$. Si i est le plus petit élément de $\mathcal{A} \backslash X$, selon l'ordre lectique, tel que $\max_{<_{lex}}(h(Inc(X, i)) \backslash X) < i$, alors $h(Inc(X, i))$ est le plus petit fermé succédant directement à X, selon l'ordre lectique.

Alg. 16 Next-Closure

Entrée : ensemble d'items $\mathcal{A}$, opérateur de fermeture h sur $\mathcal{A}$
Sortie : CL ensemble des fermés

1: $X := h(\emptyset)$
2: $CL := X$
3: **tant que** $X \neq \mathcal{A}$ **faire**
4: lower := faux
5: $S := \mathcal{A} \backslash X$
6: **tant que** $\neg$ lower **faire**
7: $i := min(S)$
8: $S := S \backslash i$
9: $X^* := h(Inc(X, i))$
10: **si** $\max_{lex}(X^* \backslash X) \leq i$ **alors**
11: lower := vrai
12: $CL := CL \cup X^*$
13: $X := X^*$
14: **fin si**
15: **fin tant que**
16: **fin tant que**
17: **return** CL

Exemple 5.23 Avec la relation binaire de l'exemple 5.17, voici une trace de l'algorithme Next-Closure :

X	S	i	$Inc(X,i)$	X^*	$T = max(X^* \backslash X)$	lower
$\emptyset$	$ABCDE$	A	A	AC	C	non
		B	B	BE	E	non
		C	C	C	C	oui
C	$ABDE$	A	AC	AC	A	oui
AC	BDE	B	BC	BCE	E	non
		D	D	ACD	D	oui
ACD	BE	B	BCD	$ABCDE$	E	non
		E	E	BE	E	oui
BE	ACD	A	ABE	$ABCE$	C	non
		C	CE	BCE	C	oui
BCE	AD	A	$ABCE$	$ABCE$	A	oui
$ABCE$	D	D	DE	$ABCDE$	D	oui
$ABCDE$						

Exercice 5.7
En considérant la relation binaire de l'exercice 5.6, utiliser l'algorithme Next-Closure pour trouver l'ensemble des fermés.

Bibliographie

[1] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast Discovery of Association Rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328, 1996.

[2] Marc Barbut and Bernard Monjardet. *Ordre et Classification*, volume 1. Hachette Université, 1970.

[3] Yves Bastide. *Data mining : algorithmes par niveau, tecniques d'implantation et applications.* PhD thesis, Université Blaise Pascal, décembre 2000.

[4] Garrett Birkhoff. *Lattice Theory*, volume XXV of *AMS Colloquium Publications.* American Mathematical Society, third (new) edition, 1970.

[5] Alain Casali. *Treillis cubes et fouilles de bases de données multidimensionnelles (à paraître).* PhD thesis, Université de la Méditerranée, octobre 2004.

[6] Alain Casali, Rosine Cicchetti, and Lotfi Lakhal. Cube Lattices: a Framework for Multidimensional Data Mining. In *Proceedings of the 3rd SIAM International Conference on Data Mining, SDM*, pages 304–308, 2003.

[7] Alain Casali, Rosine Cicchetti, and Lotfi Lakhal. Extracting semantics from data cubes using cube transversals and closures. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining, KDD*, pages 69–78, 2003.

[8] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge Mathematical Textbooks, 1990.

[9] Thomas Eiter and Georg Gottlob. Identifying The Minimal Transversals of a Hypergraph and Related Problems. In *SIAM Journal on Computing*, volume 24(6), pages 1278–1304, 1995.

[10] Christian Ernst. *Base de données, Deuxième patrie : Mise en œuvre des SGBD relationnel*. Groupe ESIM, 1999.

[11] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1999.

[12] Georg Gottlob. Computing covers for embedded functional dependencies. In *Proceedings of the 6th Symposium on Principles of Database Systems, PODS*, pages 58–69, 1987.

[13] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *Data Mining and Knowledge Discovery*, volume 1(1), pages 29–53, 1997.

[14] Haym Hirsh. Theoretical Underpinnings of Version Spaces. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence, IJCAI*, pages 665–670, 1991.

[15] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. Efficient discovery of functional and approximate dependencies using partitions. In *Proceedings of the 14th International Conference on Data Engineering , ICDE*, pages 392–401, 1998.

[16] Wenmin Li, Jiawei Han, and Jian Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM*, pages 369–376, 2001.

[17] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, KDD*, pages 80–86, 1998.

[18] Stéphane Lopes. *Data mining : algorithmes pour l'analyse de schémas de bases de données relationnelles.* PhD thesis, Université Blaise Pascal, décembre 2000.

[19] Heikki Mannila and Kari-Jouko Räihä. Design by example: An application of armstrong relations. In *Journal of Computer System Science*, volume 33(2), pages 243–250, 1986.

[20] Heikki Mannila and Kari-Jouko Räihä. *The Design of Relational Databases.* Addison Wesley, 1994.

[21] Tom M. Mitchell. Generalization as Search. In *Artificial Intelligence*, volume 18(2), pages 203–226, 1982.

[22] Tom M. Mitchell. *Machine learning.* MacGraw-Hill Series in Computer Science, 1997.

[23] E.M. Norris. An algorithm for computing the maximal rectangles in a binary relation. In *Revue Roumaine de Mathematiques Pares et Appliquees*, volume 23(2), pages 126–141, 1978.

[24] Noël Novelli. *Extraction de Dépendances Fonctionnelles dans les Bases de Données : une Approche Data Mining.* PhD thesis, Université de la Méditerranée, décembre 2000.

[25] Nicolas Pasquier. *Data mining : Algorithmes d'extraction et de réduction des règles d'association dans les bases de données.* PhD thesis, Université Blaise Pascal, janvier 2000.

[26] Antonio M. Silva and Michel A. Melkanoff. A method for helping discover the dependencies of a relation. In *Advances in Data Base Theory*, pages 115–133, 1979.

[27] Rafik Taouil. *Algorithmique du treillis des fermés : application à l'analyse formelle de concepts et aux bases de données.* PhD thesis, Université Blaise Pascal, janvier 1999.

[28] Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining, KDD*, pages 236–245, 2003.

[29] Mohammed Javeed Zaki and Karam Gouda. Fast vertical mining using diffsets. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining, KDD*, pages 326–335, 2003.

Index

$\bigvee$, 49
$\bigwedge$, 49

algorithme par niveaux, 4
Armstrong
    aximoes, 45
    relation, 61

base
    canonique, 46, 57, 60
    informative approximative, 29
    informative exacte, 28

classe d'équivalence, 12, 24, 34
classification bayesienne simple, 37
clé, 13, 25, 47
    minimale d'une relation, 63
confiance, 18, 28, 37
contrainte
    antimonotone, 6, 34
    monotone, 5, 34, 54
couverture, 23, 29, 46

dépendances fonctionnelles, 44, 53, 56, 58

ensembles
    en accord, 59–61
    maximaux, 52, 56, 57, 60, 63, 64
espace convexe, 34
espace de versions, 33

famille génératrice, 51, 53
fermeture, 34
    d'un ensemble de DF, 46
    opérateur, 13, 47
    système, 51
fermé, 24, 29, 47
forme normale
    3NF, 65
    BCNF, 64
fréquence, 7, 14, 18, 34, 37

inclusion, 13

opérateur
    semi-union, 4
ordre
    inclusion, 28, 34, 49, 54

relation
    base de données, 44
règle d'association, 18, 28

transversaux d'un hypergraphe, 5, 34, 54, 63
treillis
    de Galois, 49
    des fermés, 49
    des motifs fréquents, 10

Table des matières

I Fouille de base de données binaires **1**

1 Règles d'association **3**

1.1 Introduction aux algorithmes par niveaux 4

1.1.1 Génération des candidats pour un algorithme par niveaux 4

1.1.2 Algorithme par niveaux et contrainte monotone 5

1.1.3 Algorithme par niveaux et contrainte antimonotone . . 6

1.2 Génération des motifs fréquents 7

1.2.1 Approche *Apriori* 7

1.2.2 Approche *Pascal* 11

1.2.3 Résultats expérimentaux 17

1.3 Extraction des règles d'association 18

1.4 Close : une alternative pour l'extraction des motifs fréquents . 23

1.4.1 Résultats expérimentaux 27

1.5 Extraction de règles d'association informatives 28

1.5.1 Base informative exacte 28

1.5.2 Base informative approximative 29
1.5.3 Dérivation de la confiance d'une règle à partir de la base informative 30
1.5.4 Résultats expérimentaux 31

2 Classification **33**
2.1 Espace de Versions 33
2.1.1 Le classeur 34
2.1.2 Le classifieur 36
2.1.3 Requête SQL permettant de calculer les tuples consistants 36
2.2 Classification Bayesienne Simple 37
2.2.1 Le classeur 37
2.2.2 Le classifieur 37

II Fouille de base de données n-aires **41**

3 Dépendances Fonctionnelles **43**
3.1 Rappels sur les dépendances fonctionnelles 44
3.1.1 Un peu de vocabulaire 44
3.1.2 Dépendances fonctionnelles, bases et couvertures 46
3.1.3 Dépendances fonctionnelles et fermeture 47
3.2 Ensembles maximaux et dépendances fonctionnelles 52
3.2.1 Les ensembles maximaux 52
3.2.2 Transversaux d'un hypergraphe 54
3.2.3 Découverte des minimaux transversaux 54
3.2.4 Des ensembles maximaux aux dépendances fonctionnelles 56
3.3 Inférence des dépendances fonctionnelles 58
3.4 Relation d'Armstrong 61
3.5 Clés minimales d'une relation 63
3.6 Tests de formes normales 64
3.6.1 Test pour $BCNF$ 64
3.6.2 Test pour $3NF$ 65

4 Conclusion **69**

III Annexes 73

5 Annexe 75
5.1 Les ensembles ordonnés 75
5.1.1 Qu'est ce qu'un ensemble ordonné ? 75
5.1.2 Représentation graphique 77
5.1.3 Bornes dans un ordre 78
5.1.4 Ordres utiles pour le data mining 80
5.1.5 Contraintes et Espaces convexes 82
5.2 Notions de Treillis 84
5.2.1 Principe de dualité et treillis 85
5.2.2 Éléments irréductibles dans un trellis 86
5.2.3 Fermeture et treillis 88
5.2.4 Treillis de Galois (des fermés) 89
5.3 Calcul des fermés d'un treillis 92
5.3.1 Méthode de Norris 92
5.3.2 Algorithme Next-Closure 94

Bibliographie 97

Index 101