Université Henri Poincaré – Nancy I

Fouille de données textuelles complexes

MÉMOIRE

soutenu le 21 Juin 2004

pour l'obtention du

DEA de l'Université Henri Poincaré – Nancy I

(Spécialité Informatique)

par

Rokia Bendaoud

Composition du jury

	Noëlle Carbonell	Professeur UHP-Nancy1
	Didier Galmiche	Professeur UHP-Nancy1
	Olivier Festor	Directeur de recherche INRIA
	Dominique Mery	Professeur UHP-Nancy1 et ESIAL
Encadrants :	Amedeo Napoli	Directeur de recherche CNRS
	Yannick Toussaint	Chargé de recherche (CR1) au CNRS

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Remerciements

Ce travail a été réalisé sous la direction de Monsieur A.Napoli chef du projet ORPAILLEUR, je le remercie de m'avoir permis de faire mes premiers pas dans la recherche. Qu'il trouve ici l'expression de ma profonde gratitude.

Je suis particulièrement reconnaissante à mon encadrant Monsieur Y.Toussaint qui n'a cessé, durant ce stage, de me prodiguer conseils, remarques et encouragements.

J'adresse aussi mes remerciements à toute l'équipe ORPAILLEUR.

Je remercie également Madame N.Carbonell, Monsieur D.Galmiche et Monsieur D.Mery professeurs à UHP-Nancy1 et Monsieur O.Festor directeur de recherche INRIA pour avoir accepté de faire partie de ce jury.

Je dédie ce travail
à mes parents avec un grand Merci,
à mes frères et à ma soeur avec toute mon affection.
à mes deux petits choux : Chakibou et Momoh.
à mes amis et surtout à ma traductrice MayaBee.

Résumé

L'extraction de connaissances à partir d'une base de textes a pour but de partir de textes bruts pour arriver à des connaissances évaluées par un analyste. Notre travail se situe plus particulièrement au niveau de la fouille de données qui est une étape du processus d'extraction de connaissances à partir d'une base de textes (ECT). L'objectif de la fouille de données est d'extraire les liens entre les textes et les termes qui les composent. L'extraction de règles d'association est une méthode de fouille, qui nous permet de faire émerger tous ces différents liens, mais lorsque la base de textes est volumineuse, le nombre de règles devient difficilement gérable.

Notre travail consiste à trouver une classification des règles d'association d'après les propriétés qui les composent. Nous avons choisi pour classifier les règles la relation de subsomption, ce qui nous permet de fournir à l'expert une hiérarchie de règles facilitant leur interprétation et leur évaluation.

Mots-clés: fouille de données, règles d'association, treillis de Galois, motifs fréquents, subsomption.

Abstract

Knowledge Discovery in Texts base starts from texts and produces knowledge under the control of the analyst. Our work is in the data mining step which is one stage of the Knowledge Discovery process. In data mining, it is important to extract links between texts and between terms composing those texts. Association rules are a mining method, which allows the extraction of all those different links, but when the text database is voluminous, the number of rules becomes difficult to manage.

Our work consists in finding a classification between the different association rules according to the properties involved in these rules. We chose the relation of subsomption to classify the association rules. This relation provides the expert with a hierarchy of rules which facilitates their interpretation and evaluation.

Keywords: data mining, association rules, lattice of galois, frequents itemsets, subsomption

Table des matières

Table des figures

Liste des tableaux

Introduction

La fouille de textes (FdT) relève de l'extraction de connaissances dans les bases de données. Elle consiste à extraire des éléments dans une collection de textes qui peuvent être interprétés comme des éléments de connaissances par un expert du domaine. La fouille de textes se pratique sur des textes préparés où ont été isolés des expressions ou des termes reflétant le contenu des textes. Dans la présente approche, nous cherchons à extraire des règles d'association à partir de textes, pour les présenter ensuite à un expert pour interprétation et validation.

Le volume des règles de production extraites à partir de textes peut être très grand, et difficile à appréhender dans sa globalité par un expert humain. Nous avons donc cherché à mettre au point une relation de subsomption permettant de classifier les règles extraites par niveaux de généralités. De plus, la relation de subsomption peut également tirer parti d'une organisation hiérarchique des éléments présents dans les règles : c'est un pas supplémentaire vers une méthodologie de fouille de textes guidée par les connaissances du domaine.

Organisation du rapport

Le premier chapitre de ce mémoire décrit le processus global d'extraction de connaissances dans des bases de données, sur lequel se calque le processus d'extraction de connaissances à partir de textes, puis il situe l'étape qui concerne notre travail sur la fouille de textes. Il définit les règles d'association, et présente des indices statistiques qui permettent de classifier ces règles.

Les différentes méthodes d'extraction des règles, sont présentées dans le deuxième chapitre, qui inclut aussi la définition de la relation de subsomption que nous avons choisie pour la classification des règles.

Dans le troisième chapitre, nous définissons deux sortes de subsomption pour les règles d'association : une subsomption pour des règles avec des propriétés non hiérarchisées et une subsomption pour des règles avec des propriétés hiérarchisées. Nous présentons aussi deux exemples pour illustrer notre approche. Enfin, le quatrième chapitre conclut le mémoire.

Chapitre 1

Les règles d'association

1.1 Extraction de Connaissances à partir de Bases de Données

L'extraction de connaissances à partir de bases de données (ECBD) désigne le processus global de découverte de connaissances qui permet de passer de données brutes à des connaissances. Le processus d'ECBD se compose de plusieurs étapes, parmi lesquelles [Fayyad et al., 1996] :

- La sélection : création d'un corpus à étudier ;
- Le prétraitement : le but de cette étape est d'éliminer le bruit et de combler les manques dans les données ;
- La transformation : recherche les meilleures structures pour représenter les données en fonction des outils de fouille de données ;
- La fouille de données : la fouille proprement dite et la définition des tâches de classification et de recherches de modèles ainsi que la définition des paramètres appropriés ;
- L'interprétation et l'évaluation : pendant laquelle les unités extraites sont analysées. Les unités extraites peuvent alors après validation par un analyste être stockées dans une base de connaissances.

1.2 L'extraction de connaissance à partir d'une base de textes

Dans le cadre de la veille technologique, nous supposons que nos outils sont dédiés à un analyste, expert du domaine en charge de cette activité de veille. L'extraction de connaissances à partir de textes présente un certain nombre de spécificités par rapport à l'ECBD [Toussaint, 2004] les analyse et souligne notamment les problèmes d'extraction de l'information dans les textes et de structuration de l'information pour les outils de fouille.

Le processus d'Extraction de Connaissances à partir de Textes "ECT" suit les mêmes étapes que celui de l'ECBD, avec un prétraitement supplémentaire. La FIG.1.1 adaptée

de [Fayyad et al., 1996] présente ce processus. Nous donnons enfin la définition d'ECT proposée dans [Toussaint, 2004].

1.3 La fouille de textes

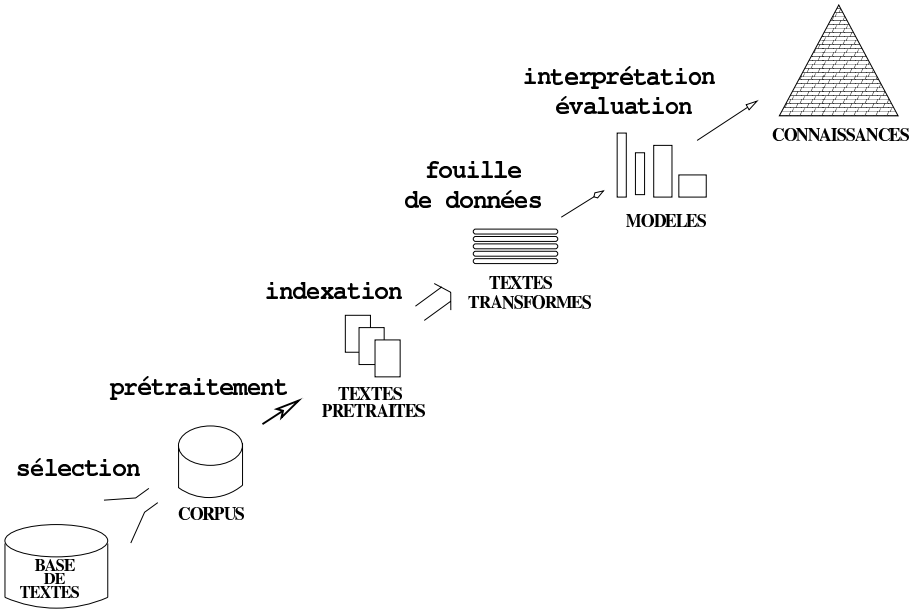

FIG. 1.1 – **Schéma global de l'ECT**

Définition 1 (L'extraction de connaissances à partir de textes)
L'extraction de connaissances à partir de textes est un processus non trivial qui construit un modèle de connaissances valide, nouveau, potentiellement utile et au final compréhensible, à partir de textes bruts.

Nous nous intéressons plus particulièrement à l'étape de fouille de textes qui correspond sur le schéma à celle de fouille de données. La méthode de fouille que nous utilisons est l'extraction de règles d'association. La lisibilité des règles d'association en fait une méthode de fouille attractive mais le très grand nombre de règles constitue un point négatif, car cet ensemble devient difficilement exploitable par l'analyste. L'idée pour faciliter l'analyse est d'ordonner et de structurer cet ensemble. Nous allons considérer les textes comme des individus et les termes qui en sont extraits comme des propriétés pour en obtenir un tableau booléen entre ces deux ensembles.

1.4 Les règles d'association

Une règle d'association est du type R : A $\Rightarrow$ B, où A et B sont des ensembles de propriétés.

Définition 2 (Les règles d'association)
Soit P l'ensemble des propriétés. Une règle d'association est une implication de la forme $A \Rightarrow B$, ou A est la prémisse (body), et B est la conclusion (head) avec $A \subseteq P$, $B \subseteq P$ et $A \cap B = \emptyset$.

Les règles d'association permettent de mettre en évidence les dépendances entre les propriétés. Par exemple, la règle : `vole` => `pond` $\wedge$ `respire` (la conjonction de propriétés est notée comme en logique) peut s'interpréter comme le fait que si un individu `vole`, il est probable qu'il ait la propriété `pond` et `respire`. Lorsque nous avons plusieurs propriétés, le nombre de règles augmente, ce qui devient difficilement gérable par l'analyste. Des indices statistiques ont été définis pour tenter d'ordonner les règles des plus pertinentes aux moins pertinentes pour l'analyste.

1.4.1 Les indices statistiques

Les indices statistiques (support et confiance) servent à limiter le nombre de règles d'association extraites et certains d'autres indices ont été définis pour ordonner les règles d'association, en fonction de critères statistiques. Il existe plusieurs indices dont les deux les plus utilisés sont :

Le support

Cet indice représente le nombre d'individus qui possèdent les deux ensembles de propriétés A et B.

$$Support(A \Rightarrow B) = support(A \cup B)$$

Pour limiter le nombre de règles, on définit un seuil appelé *minsupp*, au dessous duquel les règles sont éliminées.

La confiance

Cet indice peut s'interpréter pour la règle A $\Rightarrow$ B comme la probabilité conditionnelle Prob(B|A) qui est la probabilité qu'un individu possède les propriétés B sachant qu'il possède celles de A [Agrawal and Srikant, 1995] :

$$confiance(A \Rightarrow B) = \frac{Support(A \Rightarrow B)}{Support(A)}$$

Pour limiter le nombre de règles, on définit un seuil appelé *minconf*, au-dessous duquel les règles sont non valides. Ces deux indices servent à réduire la complexité du processus d'extraction de règles comme nous allons le voir dans la section 2.2

Définition 3 (Règle valide)
Une règle $R : A \Rightarrow B$ est valide si support (R)$\geq$ minsupp et donc la confiance (R) $\geq$ minconf.

Autres indices

Nous présentons d'autres indices, détaillés dans [Cherfi et al., 2003], qui permettent différents classements des règles. Nous définissons la probabilité d'un motif par la proportion de ce motif dans les textes. Soit A $\subset$ P

$$P(A) = \frac{D(A)}{D}$$

où D(A) est l'ensemble des documents contenant le motif A et D l'ensemble total des documents. Nous définissons également P(A $\cup$ B) par la probabilité du motif A $\cup$ B.

- L'intérêt mesure la déviation du support de la règle par rapport au cas d'indépendance. Rappelons que deux événements A et B sont indépendants si :
  support (A $\Rightarrow$ B) = P(A $\cup$ B) = P(A)$\times$P(B).

  $$intérêt(A \Rightarrow B) = \frac{P(A \cup B)}{P(A) \times P(B)}$$

  Plus A et B sont dépendants, plus l'intérêt de la règle R est supérieur à 1.
- La conviction mesure la déviation du support du contre-exemple à la règle A $\cap \neg$B par rapport à l'indépendance de A et $\neg$B.

  $$conviction(A \Rightarrow B) = \frac{P(A) \times P(\neg B)}{P(A, \neg B)}$$

  La conviction n'est pas symétrique, et dénote une dépendance entre A et B lorsqu'il est $>$1, une indépendance lorsqu'il est $= 1$ et pas de dépendance s'il est compris entre [0,1[.
- La dépendance est utilisée pour mesurer une distance de confiance de la règle par rapport au cas d'indépendance.

  $$dépendance(A \Rightarrow B) = |P(B|A) - P(B)|$$

  Plus cet indice est proche de 0 (resp de 1), plus A et B sont indépendants (resp dépendants) .

1.4.2 Les règles totales

Les règles totales sont les règles dont la une confiance est égale à 1, cela signifie qu'à chaque fois que A apparaît dans un texte, B en découle nécessairement. Les règles totales ne possèdent donc pas de contre-exemples.

1.4.3 Les règles partielles

Les règles partielles sont les règles dont la confiance est < 1. Ce sont donc des règles qui possèdent des contre-exemples qui vérifient la partie gauche de la règle mais pas la partie droite.

1.4.4 Les règles informatives

Une règle R : A $\Rightarrow$ B est dite informative si et seulement si elle est valide et : A $\cap$ B $= \oslash$

1.4.5 Propriétés des règles

- transitivité : si A $\Rightarrow$ B et B $\Rightarrow$ C alors A $\Rightarrow$ C.
- si A $\Rightarrow$ B alors $\forall$ X $\subseteq$ P, A $\Rightarrow$ B $\sqcup$ X (B $\sqcup$ X représente un ensemble de propriétés plus général, ou, en terme de motifs, un motif plus court).

Ce chapitre nous a permis de situer le cadre de notre sujet, de presenter la méthode de fouille que nous avons choisie qui est l'extraction de règles d'association puis de donner les propriétés de cette méthode qui vont nous servir de support pour le prochain chapitre. Nous montrons dans la suite de ce rapport l'importance de classifier et de structurer les règles d'association pour pouvoir les interpréter et les manipuler plus facilement.

Chapitre 2

L'extraction de règles d'association et la subsomption

Introduction

Pour pouvoir structurer et hiérarchiser les règles d'association, nous allons d'abord présenter les différentes méthodes d'extraction de règles, puis nous présentons la relation de subsomption avec deux point de vue différents qui vont nous permettre de définir une subsomption entre règles.

2.1 L'extraction de règles à partir d'un treillis de Galois

Un treillis de Galois permet d'obtenir des classes structurées hiérarchiquement qui regroupent les individus partageant les mêmes propriétés. Plusieurs travaux font référence aux treillis de galois, entre autres [Guénoche, 1990] [Barbut and Monjardet, 1970] [Ganter and Wille, 1999] ou sur des données complexes [Polaillon, 1998].

Définition 4 (Contexte formel)
Un concept formel est un triplet K=(I,P,R) où I est un ensemble d'individus, P est un ensemble de propriétés et R une relation binaire entre I et P vérifiant :
- *$R \subseteq I \times P$.*
- *(i,p) $\in$ R avec $i \in I$ et $p \in P$ signifie que l'individu i possède la propriété p ou que la propriété p est possédée par l'individu i.*

Connexion de Galois

Soit $\mathcal{P}$(I) (respectivement $\mathcal{P}$(P)) l'ensemble des parties de I (respectivement P). Considérons les deux applications suivantes :

- f: $\mathcal{P}$(I) $\rightarrow$ $\mathcal{P}$(P) tel que f (X)= {p $\in$ P | $\forall$ i $\in$ X, iRp}.
  f associe à chaque ensemble d'individus de I l'ensemble des propriétés qu'ils ont en commun dans P.

- $g : \mathcal{P}(\mathrm{P}) \rightarrow \mathcal{P}(\mathrm{I})$ tel que g (Y)= {i ∈ I | ∀ p ∈ Y, iRp}.
  g associe à chaque ensemble de propriétés de P, l'ensemble des individus qui les possèdent dans I.

Les applications f et g sont décroissantes et elles possèdent les propriétés suivantes :
- ∀ (I_1,I_2) ∈ $\mathcal{P}$(I), $I_1 \subseteq I_2 \Rightarrow f(I_2) \subseteq f(I_1)$
- ∀ (P_1,P_2) ∈ $\mathcal{P}$(P), $P_1 \subseteq P_2 \Rightarrow g(P_2) \subseteq g(P_1)$
- ∀ P_1 ∈ $\mathcal{P}$(P), $P_1 \subseteq f(g(P_1))$ et ∀ I_1 ∈ $\mathcal{P}$ (I), $I_1 \subseteq g(f(I_1))$

Définition 5 (La connexion de Galois)
*Le couple (*f, g*) forme une connexion de Galois entre ($\mathcal{P}$(I),⊆) et ($\mathcal{P}$(P),⊆).*

Fermeture

Une fermeture dans un ensemble ordonné (E, ≤) est une application, H : E → E, telle que pour tout (x,y) ∈ E, les propriétés suivantes sont vérifiées :
- H est extensive : x ≤ H(x).
- H est monotone croissante : x ≤ y ⇒ H(x) ≤ H(y)
- H est idempotente : H(x) = H(H(x))

Définition 6 (Fermé)
Un sous ensemble X de E est dit fermé pour l'opérateur de fermeture H si et seulement si : X = H(X).

Les applications h=$f \circ g$ et h'= $g \circ f$ sont respectivement des fermetures sur ($\mathcal{P}$(I),⊆) et et ($\mathcal{P}$(P),⊆).

Définition 7 (Treillis)
Un ensemble ordonné (E,≤) est un treillis, *si et seulement si tout couple d'éléments (x,y) de E possède une borne supérieure notée : $x \vee y$, et une borne inférieure notée : $x \wedge y$, unique. $x \leq x \vee y$ et $y \leq x \vee y$ et il n'existe aucun élément z tel que : $x \leq z \leq x \vee y$ et $y \leq z \leq x \vee y$.*

Soient F_I et F_P les ensembles des fermés respectifs pour h et h'. (F_I,⊆) est le treillis des fermés pour h et (F_P,⊆) est le treillis des fermés pour h'. (F_P,⊆) et (F_I,⊆) sont deux treillis isomorphes

Concept formel

Pour tout $I_1 \subseteq$ I, on définit P_1 tel que :
$P_1 = f(I_1)$ = {p ∈ P | ∀ i ∈ I_1, iRp} et pour tout $P_1 \subseteq$ P on définit I_1 tel que : $I_1 = f(P_1)$ = {i ∈ I | ∀ p ∈ P_1, iRp}. Un concept formel ayant pour contexte formel (I,P,R) est défini comme un couple C=(I_1,P_1) avec $I_1 \subseteq$ I et $P_1 \subseteq$ P. Les ensembles I_1 et P_1 sont appelés respectivement *extension* et *intension* du concept formel C.

2.1.1 Treillis de Galois

Soient C1 = (I_1,P_1) $\preceq$ C2 = (I_2,P_2), et $\preceq$ la relation de subsomption entre concepts formels : C1 $\preceq$ C2 $\Leftrightarrow$ $I_1 \subseteq I_2$ et $P_2 \subseteq P_1$.
On dit que C1 est subsumé par C2 et que C2 est le subsumant de C1.

Définition 8 (Treillis de Galois)
Soit (F,$\preceq$) le produit des deux treillis (F_I,$\preceq$) et (F_P,$\preceq$) appelés respectivement treillis des extensions et treillis des intensions. (F,$\preceq$) est le treillis de Galois *associé à la relation binaire R sur I×P. L'ensemble de tous les concepts formels du contexte K=(I,P,R) muni de l'ordre partiel $\preceq$ est un treillis appelé* treillis des concepts *de K ou* treillis de Galois

On présente un exemple dans le tableau 2.1 d'une relation R entre 5 individus et de 9 propriétés, la figure FIG.2.1 représente le diagramme de Hasse représentant la relation R [Godin et al., 1995].

R	a	b	c	d	e	f	g	h	i
1	1	0	1	0	0	1	0	1	0
2	1	0	1	0	0	0	1	0	1
3	1	0	0	1	0	0	1	0	1
4	0	1	1	0	0	1	0	1	0
5	0	1	0	0	1	0	1	0	0

TAB. 2.1 – Représentation en tableau de la relation R

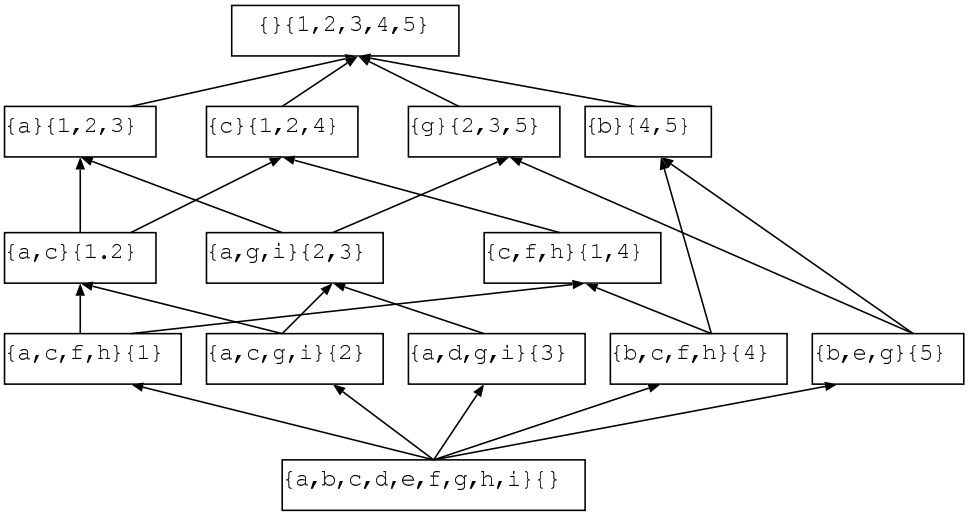

FIG. 2.1 – **Diagramme de Hasse**

2.1.2 Méthodes de construction des treillis de Galois

1. Algorithmes non incrémentaux : ces algorithmes obligent à reconstruire le treillis à chaque fois que l'ensemble des individus ou que l'ensemble des propriétés sont modifiés. Ces algorithmes sont extraits de [Guénoche, 1990] :
   - Norris : recherche des rectangles maximaux en examinant le tableau ligne par ligne
   - Bordat : construction simultanée des fermés et du diagramme de Hasse par recherche des fermés couverts par un élément du treillis
   - Chein : recherche des rectangles maximaux par opération d'union sur les extensions et d'intersection sur les intensions
   - Ganter : construction des fermés pour une relation de fermeture avec utilisation de l'ordre lexicographique

   Une comparaison de ces quatre algorithmes est donnée dans [Guénoche, 1990]. L'algorithme de Ganter est le plus efficace pour des données de plus de 15 individus. La méthode de Bordat est la seule à construire directement le diagramme de Hasse du treillis. La méthode de Norris est la plus rapide et celle de Chein est la plus facile à simuler à la main.
2. Algorithmes incrementaux
   Si l'ensemble des propriétés ou des individus augmentent, les algorithmes incrémentaux permettent de modifier localement les noeuds qui doivent être modifiés et d'insérer éventuellement de nouveaux noeuds [Godin and Missaoui, 1994].

2.1.3 Méthodes d'extraction de règles dans les treillis

La structure du treillis de Galois peut être exploitée afin d'extraire des règles d'association valides [Simon, 2000], qu'elles soient partielles ou totales. Le treillis de Galois permet de définir un ensemble minimal de règles appelé *base canonique* à partir de quelle toutes les règles informatives peuvent être inférées par applications des trois règles suivantes :

1. transitivité : A $\Rightarrow$ B est une conséquence de A $\Rightarrow$ D et de D $\Rightarrow$ B.
2. A $\Rightarrow$ B est vrai dès que B $\subseteq$ A
3. A $\cup$ X $\Rightarrow$ B $\cup$ X est une conséquence de A $\Rightarrow$ B

2.1.4 L'héritage dans le treillis de Galois

L'héritage dans le treillis de Galois [Godin et al., 1995] permet de distinguer les propriétés propres (resp. individus) des propriétés héritées. Grâce à la relation de subsomption dans le treillis de galois : C1 = (I_1,P_1) $\preceq$ C2 = (I_2,P_2) $\Leftrightarrow$ $I_1 \subseteq I_2 \wedge P_2 \subseteq P_1$.

Nous pouvons considérer que c'est une répétition de réécrire les individus de C1 dans C2, donc nous allons ôter les individus de C2 qui existe dans C1 et définir une variante d'écriture qui symbolise l'héritage dans un treillis de galois, appelée treillis d'héritage

selon les individus. Nous pouvons déduite la même conclusion pour les propriétés, et ôter les propriétés de C1 qui existent dans C2, le treillis résultant est un treillis d'héritage selon les propriétés [Godin et al., 1995].

- L'héritage dans le treillis de Galois par rapport à l'ensemble des individus "I" :
  La figure FIG.2.2 montre l'héritage dans le treillis de la figure FIG.2.1 par rapport à l'ensemble des individus. Un individus est possédé par tous les ascendants du concept où il apparaît.

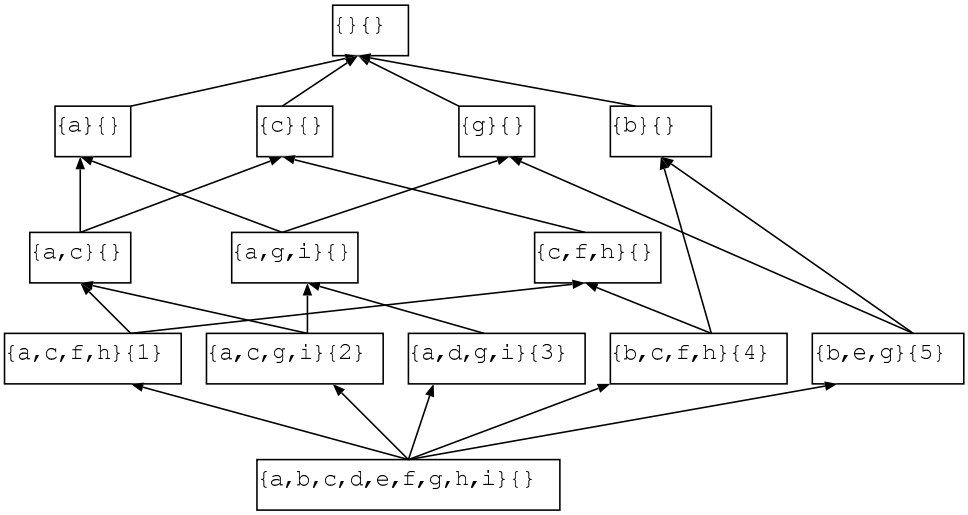

FIG. 2.2 – **L'héritage dans le treillis des individus**

- l'héritage dans le treillis de Galois par rapport à l'ensemble des propriétés "P"
  La figure FIG.2.3 montre l'héritage dans le treillis de la figure FIG.2.1 par rapport à l'ensemble des propriétés. Une propriété est possédée par tous les descendants du concept où elle apparaît.
- L'héritage dans le treillis de Galois par rapport aux ensembles "I et P"
  La figure FIG.2.4 montre l'héritage du treillis de la figure FIG.2.1 par rapport à l'ensemble des individus et à l'ensemble des propriétés.

2.1.5 L'élagage du treillis d'héritage

L'élagage d'un treillis par rapport aux individus (respectivement par rapport aux propriétés) consiste à supprimer les concepts pour lesquels l'extension (respectivement l'intension) est vide. Dans la figure 2.5, nous présentons le résultat de l'élagage selon les intensions du treillis de la figure 2.3

2.1.6 Extraction des règles d'un treillis d'héritage

Une base de règles peut être obtenue à partir d'un treillis d'héritage [Simon, 2000]. Pour chaque concept, nous pouvons déduire d'une part l'ensemble des propriétés "propres"

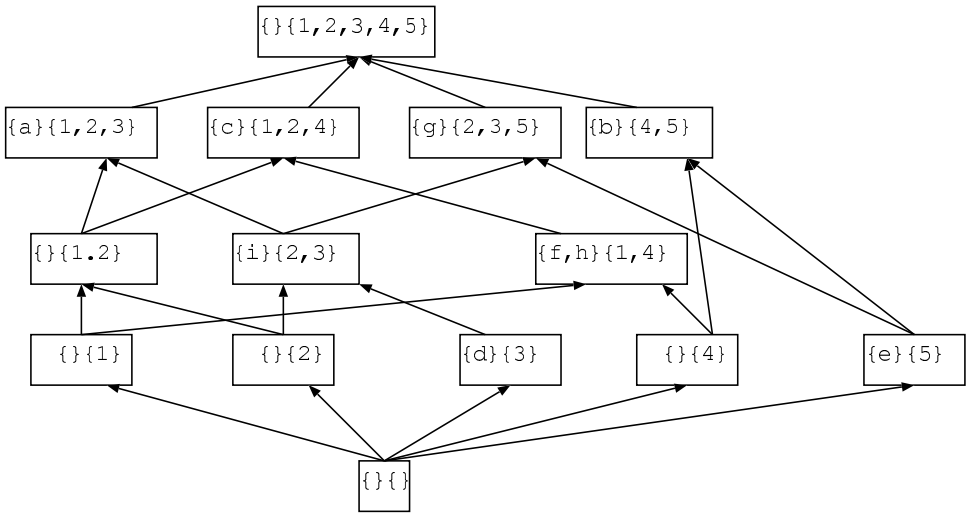

FIG. 2.3 – **L'héritage dans le treillis des propriétés**

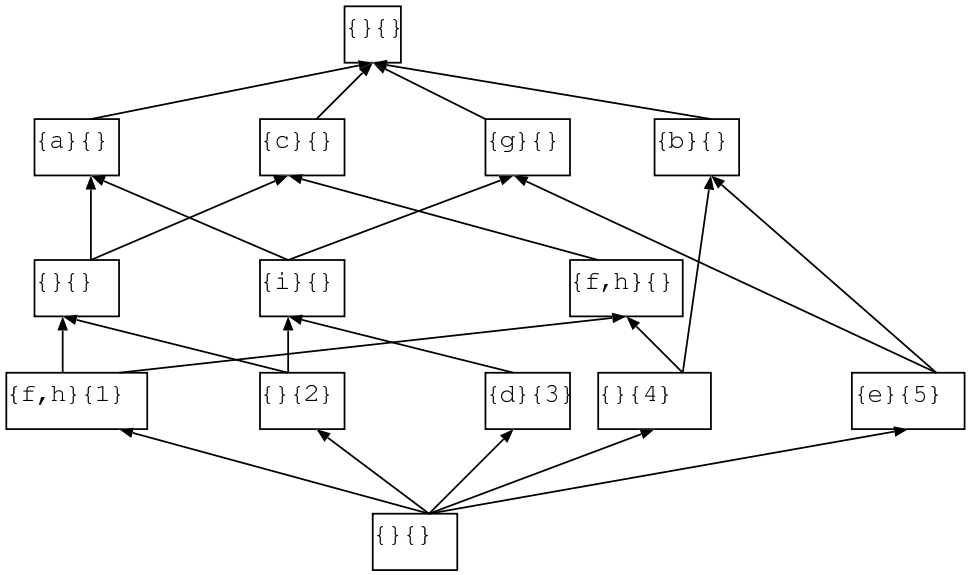

FIG. 2.4 – **L'héritage dans le treillis des individus et des propriétés**

et d'autre part l'ensemble des propriétés "héritées". Les propriétés "propres" sont celles qui figurent dans le treillis d'héritage selon l'intension. Chaque concept permet alors de générer les règles d'implication : $p_i \Rightarrow p_h$ où $p_i \in$ propres et $p_h \in$ héritées. Les propriétés propres sont mutuellement équivalentes lorsqu'elles sont dans un même concept. Par exemple, du treillis FIG.2.5, nous déduisons à partir du concept C7, les règles suivantes :

- f $\Rightarrow$ c
- h $\Rightarrow$ c.
- f $\Rightarrow$ h.

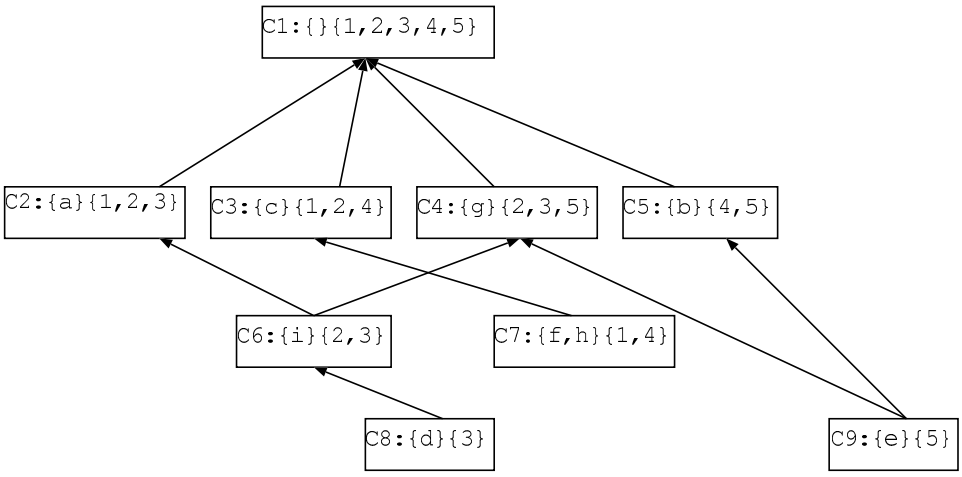

FIG. 2.5 – **L'élagage dans le treillis d'héritage des propriétés**

2.2 Extraction de règles à partir des algorithmes d'extraction de motifs fréquents

La définition du motif fréquent à été extraite de [Bastide et al., 2002]

Définition 9 (motif)
Un motif est un sous-ensemble de P. On dit qu'un individu contient le motif P_1 *, si* P_1 *et i sont en relation :* $\forall$ $p \in P_1$, $(i,p) \in R$. *Un motif de taille k est noté k-motif.*

Définition 10 (motif fréquent)
Un motif P_1 *est dit fréquent si et seulement si : support* $(P_1) \geq$ *minsupp.*

2.2.1 Extraction de règles avec de propriétés non hiérarchisées

Dans ce qui suit, nous supposons que l'ensemble des propriétés n'est pas structuré.

Algorithme Apriori

L'algorithme Apriori [Agrawal et al., 1993] calcule l'ensemble des motifs fréquents. D'abord il calcule tous les motifs fréquents de taille 1, puis les motifs de longueur supérieure. Il effectue un calcul par niveaux et supprime les motifs non fréquents, sur la base de ces deux principes :

- chaque sous-motif d'un motif fréquent est fréquent.
- chaque super-motif d'un motif non fréquent est un motif non fréquent.

Il répète ce processus jusqu'à l'obtention de tous les motifs fréquents. Pour chaque motif fréquent S, il extrait les règles d'association du type : s $\Rightarrow$ S s tel que s est un sous ensemble de S.

Algorithme Pascal

Définition 11 (relation d'équivalence)
La relation d'équivalence θ est définie comme suit :
$p\theta q$ où p et q sont des motifs équivalents si et seulement si : f*(p)* = f*(q)*, *où* f*(p) est l'extension du motif p. Tous les motifs équivalents sont dans une classe d'équivalence.*

Définition 12 (motif clé)
Un motif clé est un motif de longueur minimale dans sa classe d'équivalence.

L'algorithme Pascal : [Bastide et al., 2002]
Cet algorithme est une extension de l'algorithme Apriori. Il limite les accès à la base de données, grâce aux classes d'équivalences entre motifs.

2.2.2 Extraction de règles avec de propriétés hiérarchisées

S'il existe un ordre partiel entre les propriétés, une taxonomie, les algorithmes suivant utilisent cette hiérarchie pour extraire les motifs. Dans la hiérarchie des propriétés, s'il existe une relation de spécialisation entre P_1 et P_2, alors nous appelons P_1 ancêtre de P_2 et P_2 le descendant de P_1.

Algorithme Basic

L'algorithme Basic [Agrawal and Srikant, 1995] calcule les motifs fréquents avec la même méthode que Apriori, puis il rajoute tous les ancêtres de chaque propriété, en supprimant les redondances, et enfin recalcule les motifs fréquents.

Algorithme Cumulet

L'algorithme Cumulet [Agrawal and Srikant, 1995] est une amélioration de l'algorithme Basic, les améliorations apportées sont :
- filtrer les ancêtres : on n'ajoute pas tous les ancêtres des propriétés mais on ajoute juste les ancêtres des propriétés qui sont dans des motifs fréquents.
- pour trouver les ancêtres des propriétés, on traverse la taxonomie
- enlever les propriétés qui sont présentes elles et leurs ancêtres.

Algorithme de Maedche et Staab

Cet algorithme tiré de [Maedche and Staab, 2000a] reprend l'idée d'Agrawal dans [Agrawal and Srikant, 1995]. Il calcule les motifs fréquents. Chaque motif fréquent $t_i = (a_{i,1},a_{i,2})$ est nommé motifs pairs "MP". On dit que $(a_{i,1},a_{i,l}) \in$ H si et seulement si $a_{i,l}$ est l'ancêtre de $a_{i,1}$, X et Y sont des ensembles de motifs. On appelle $\widehat{X} \Rightarrow \widehat{Y}$ l'ancêtre de la règle X $\Rightarrow$ Y, ou $\widehat{X}$ est l'ancêtre de X et $\widehat{Y}$ l'ancêtre de Y dans la hiérarchie des propriétés. L'algorithme s'effectue en quatre étapes :

1. Déterminer T := $\{\{a_{i,1},a_{i,2},...,a_{i,m}\}|(a_{i,1},a_{i,2}) \in$ MP $\wedge$ l$\geq$3 $\rightarrow$ $((a_{i,1},a_{i,l}) \in$ H $\vee$ $(a_{i,2},a_{i,l}) \in$ H $)\}$.

2. Déterminer support de toutes les règles d'association du type X $\Rightarrow$ Y, tel que : |X| = |Y| = 1.
3. Déterminer la confiance des règles de 2 qui possèdent un support $\geq$ minsupp.
4. faire ressortir toutes les règles de 3 qui possèdent une confiance $\geq$ minconf et qui ne possèdent pas de règles ancêtres avec un support et une confiance supérieure ou égale.

Nous venons de voir que l'extraction de règles d'association n'est pas un processus simple, et il faut trouver un moyen pour utiliser au mieux les règles d'association. Une structuration des règles va nous permettre de bien exploiter toutes les possibilités d'extraire de la connaissances. Pour structurer les règles d'association, il nous faut définir une relation d'ordre, qui va s'appuyer sur des concepts [Wille, 1992] cette relation est la "subsomption".

2.3 La subsomption

La subsomption permet de définir une classification. Nous allons définir deux types de subsomption différents, la premiere pour des concepts en logique de descriptions et la deuxième pour des clauses en programmation logique inductive.

2.3.1 La subsomption en logique de descriptions

Nous présentons d'abord la subsomption en logique de descriptions "LD" tirée de [Napoli, 1997], qui nous permettra de justifier notre premiere définition de subsomption de règles avec des propriétés non hiérarchisées.

Définition 13 (Subsomption en LD)
Un Concept D est subsumé par un concept C (respectivement C subsume D) ce qui se note : D $\sqsubseteq$ C(respectivement C $\sqsupseteq$ D) si et seulement si $D^I \subseteq C^I$ pour toute interprétation I. Le concept C est appelé le subsumant et D le subsumé .

Propriétés de la subsomption en logique de descriptions

La subsomption en logique de descriptions possède les propriétés suivantes :
– la subsomption est réflexive : un concept est subsumé par lui même.
– la subsomption est transitive : si C $\sqsubseteq$ E et E. $\sqsubseteq$ D alors C $\sqsubseteq$ D.
– la subsomption est anti-symétrique : si C $\sqsubseteq$ E et E $\sqsubseteq$ C alors C=E.
– si C $\sqsubseteq$ D alors $\forall$ X concept, C $\sqcap$ X $\sqsubseteq$ D.
– si C $\sqsubseteq$ D alors $\forall$ X concept, C $\sqsubseteq$ D $\sqcup$ X.
– si C $\sqsubseteq$ D et C $\sqsubseteq$ E alors C $\sqsubseteq$ D $\sqcup$ E (propriété de treillis)

2.3.2 La subsomption en programmation logique inductive

La deuxième subsomption est la subsomption en programmation logique inductive PLI [Cornuéjols and Miclet, 2001].

La PLI réalise l'apprentissage de formules de la logique des prédicats à partir d'exemples et de contre-exemples. L'enjeu est de construire des expressions logiques comportant des variables liées les unes aux autres.

Il existe plusieurs formules en logique des prédicats, celles qui nous intéressent sont les clauses car elles sont similaires aux règles d'association. Nous allons définir ce que c'est qu'une clause pour pouvoir, dans le chapitre suivant, s'inspirer de la définition de la subsomption entre clauses et de calquer cette dernière sur les règles d'association.

Définition 14 (Clause)

Une clause est une formule de la logique des prédicats, qui se compose d'une disjonction finie de littéraux dont toutes les variables sont quantifiées universellement. Une clause s'écrit : $\neg B_1 \vee \neg B_2 \vee ... \vee \neg B_n \vee A_1 \vee A_2 .... \vee A_m$ *mais aussi en abrégé :*
$B_1, B_2, ..., B_n \rightarrow A_1, A_2, ..., A_m$

Définition 15 (Théorie)

Une théorie est un ensemble de Clause.

Définition 16 (La subsomption des théories)

Clause1 subsume Clause2 relativement à la théorie T si : $T \wedge$ *Clause1* $\models$ *Clause2, ce que nous notons :* *Clause1* $\models_T$ *Clause2*

Chapitre 3

La subsomption entre règles d'association

3.1 Problématique

Pour pallier la difficulté de lecture du très grand nombre de règles, nous avons choisi de les structurer entre elles pour permettre une lecture descendante. Cette structuration repose sur la subsomption. La présentation se fait selon deux approches : la subsomption pour les règles avec des propriétés non hiérarchisées et la subsomption pour les règles avec des propriétés hiérarchisées. Ensuite, nous analysons les informations déduites de la subsomption de règles pour chacune des définitions.

3.2 Pourquoi classifier les règles d'association

Les règles d'association permettent d'extraire des connaissances à partir d'une classification de concepts. Le fait de classifier et de structurer les règles en une hiérarchie va aider l'analyste à les évaluer et à les structurer. Le type de classification des règles dépend des propriétés des règles et de ce qu'on veut en faire. Pour le DEA, nous avons défini deux types de subsomption différents

Le premier type de subsomption est la subsomption de règles dans le cas ou les propriétés sont non hiérarchisées, aucune propriété n'est alors plus générale (ni plus spécifique) qu'une autre. Par exemple des propriétés du type : `vol`, `denté`, `pondre`... sont indépendantes les unes des autres. La classification de ce type de règles offre les avantages suivants :

- Dans certains corpus, on arrive vite à un nombre très élevé de règles, et il est difficile pour l'expert de s'y retrouver. Par exemple pour un corpus de 1361 documents (textes) d'environ 240 000 mots, en fixant le seuil minconf à 1 (règles totales) et le seuil minsupp à 0.7, on a extrait 1202 règles (voir [Cherfi and Toussaint, 2002]), ce qui est difficile à appréhender pour l'expert. Au lieu de lui fournir des indices statistiques difficiles à interpréter, nous lui fournissons une hiérarchie de règles, Le parcours de la hiérarchie se fait de haut en bas, en partant des règles avec le support

le plus élevé vers celles avec le plus petit support.
- La hiérarchie des règles va être directement déduite du treillis de galois, grâce aux propriétés du treillis.

Le deuxième type de subsomption est la subsomption de règle dans le cas où les propriétés sont hiérarchisées, comme par exemple : `QualityofService(QoS)`, `Réseaux` et `informatique`, la propriétés `QoS` est plus spécifique que la propriété `réseaux`, et la propriété `informatique` est plus générale que la propriété `réseaux`... Le fait de classifier ces règle nous offre d'autres avantages :
- Lorsqu'un analyse étudie une règle du type A $\Rightarrow$ B et qu'il trouve que cette règle est trop spécifique, il pourra généraliser l'une des deux parties de la règle, en parcourant la hiérarchie dans le sens ascendant. Par contre s'il trouve cette même règle trop générale, il pourra la spécifier avec un parcours descendant dans la hiérarchie des règles.
- Dans une base de textes, on trouve souvent des propriétés (termes) hiérarchisées, et c'est intéressant de faire apparaître une relation entre deux textes qui ne possèdent pas le même terme mais que l'un possède un terme plus spécifique que l'autre.

3.3 Subsomption des règles avec des propriétés non hiérarchisées

Pour la définition de la subsomption entre règles d'association avec des propriétés non hiérarchisées, que nous allons appelé I-subsomption (par rapport aux individus) et noté $\sqsubseteq_I$, nous nous sommes inspirés de la subsomption en logique de descriptions (voir section 2.3.1), qui s'appuie sur l'extension des concepts. Le but de cette relation subsomption est la classification des règles en s'appuyant sur les individus qui les vérifient. Cela va nous permettre de déduire la hiérarchie des règles à partir du treillis de galois dont elles sont issues.

Définition 17 (I-Subsomption des règles)
Soient deux règles R1 : A $\Rightarrow$ B et R2 : C $\Rightarrow$ D, I_1 est l'ensemble des individus qui vérifient la règle R1 appelé extension de R1 et I_2 l'ensemble des individus qui vérifient la règle R2 appelé extension de R2.
$P_1 = A \cup B$ est l'intension de R1 et $P_2 = C \cup D$ est l'intension de R2. R1 I-subsume R2 (noté :R2 $\sqsubseteq_I$ R1) si et seulement si :
$\forall\ i \in I_2 => i \in I_1$ ce qui est équivalent à $I_2 \subseteq I_1$ et $\forall\ p \in P_1 => p \in P_2$ ce qui est équivalent à $P_1 \subseteq P_2$.
En d'autres termes : si i vérifie la règle C $\Rightarrow$ D alors i possède aussi A $\Rightarrow$ B. En d'autres termes aussi : le support de R2 est inclus dans le support de R1.

3.3.1 Interprétation

La classification sur la base de l'extension, signifie que si la règle R1 subsume la règle R2 alors l'extension de R2 est incluse dans celle de R1. Nous savons que dans un treillis de galois, un concept C1 subsume C2 implique que l'extension de C2 est incluse dans

l'extension de C1 2.1.1 et donc chaque règle extraite de l'intension de C1 subsume toutes les règles extraites de l'intension de C2.

3.3.2 Les classes des règles avec des propriétés non hiérarchisées

Définition 18 (Les classes des règles avec des propriétés non hiérarchisées)
Soient deux règles R1 : A ⇒ B d'extension I1, et d'intension P1 = A ∪ B, et R2 : C ⇒ D d'extension I2, et d'intension P2=C ∪ D. Les deux règles sont de la même classe si et seulement si : R1 $\sqsubseteq_I$ R2 et R2 $\sqsubseteq_I$ R1 ⇔ I1 = I2 et P1 = P2.

Interprétation

Le fait que deux règles soient dans la même classe, signifie qu'elles possèdent la même extension et la même intension et donc qu'elles ont été extraites du même concept dans le treillis, de ce fait elles se retrouvent au même noeud dans la hiérarchie. nous ne pouvons pas remplacer une règle par une autre règle de la même classe car elles ne sont pas équivalentes.

Exemple : Soient le motif P1 = abc et I1 l'extension de ce motif, on peut extraire les règles suivantes :
⇒ abc, a ⇒ bc, b ⇒ ac, c ⇒ ab, ab ⇒ c, ac ⇒ b, bc ⇒ a. Ces règles sont de la même classe car elles ont la même intension et la même extension, mais elles ne sont pas équivalentes.

Interprétation

Le fait que deux règles soient dans la même classe, signifie qu'elles possèdent la même extension et la même intension et donc qu'elles ont été extraites du même concept dans le treillis, de ce fait elles se retrouvent au même noeud dans la hiérarchie. nous ne pouvons pas remplacer une règle par une autre règle de la même classe car elles ne sont pas équivalentes.

3.3.3 Propriétés de la I-subsomption et des classes de règles

- transitivité : si R1 $\sqsubseteq_I$ R2 et R2 $\sqsubseteq_I$R3 alors : R1 $\sqsubseteq_I$ R3
- réflexivité : R $\sqsubseteq_I$ R
- anti-symétrie : R1 $\sqsubseteq_I$ R2 et R2 $\sqsubseteq_I$ R1 alors R1 ≡ R2.
- si R1 $\sqsubseteq_I$ R2 et R2 ≡ R3 alors R1 $\sqsubseteq_I$ R3

3.3.4 Expérimentation sur la base du "zoo"

Dans cette section, nous présentons une experimentation permettant d'illustrer l'intérêt de la I-subsomption. Nous avons exploité une base de données partagée par la communauté sur de la fouille de données. Les critères imposés sont d'une part, de considérer les propriétés comme étant non hiérarchisées, et d'autre part de limiter la taille des données

pour que l'exemple soit compréhensible. Nous exploitons la base "Zoo" de [Forsyth, 1991] dont les individus sont des types d'animaux (antilope, ours, sanglier,..) et dont les propriétés (pond, vole, est-poilu,...) sont données en Annexe A.1. Nous avons réduit cette base de données à 40 individus avec 19 propriétés binaires. Nous avons construit le treillis de Galois et extrait les règles d'association à l'aide du logiciel Galicia [Valtchev et al., 2003]. Les règles d'associations (présentées dans l'annexe A.2) ont était extraites avec : minconf à 0.5 et minsupp = 0.3. Nous avons obtenu 38 règles partielles et 7 règles totales. La hiérarchie des règles selon la I-subsomption à partir du treillis de Galois, est présentée dans FIG.3.1.

Chaque noeud représente les règles valides qui ont pu être extraites à partir d'un concept du treillis, ces règles étant donc issues de la même classe car leur extension est la même.
Exemple : La règle R17 : vertébré ⇒ respire est de la même classe que la règle R20 : respire ⇒ vertébré, mais ces règles ne sont pas équivalentes. Les 4 règles suivantes sont aussi de la même classe car elles sont toutes extraites du concepts C136 :

- R21 : 4pattes ⇒ denté, respire,vertébré
- R32 : denté,respire,vertébré ⇒ 4pattes
- T5 : 4pattes,vertébré ⇒ denté,respire
- T6 : 4pattes,respire ⇒ denté,vertébré

Fournir à analyste la hiérarchie FIG.3.2 lui permet de partir de la règle R3 : ⇒ respire (cette règle veut dire l'ensemble des individus possèdent la propriété respire), si cette règle lui semble intéressante, il peut rechercher des règles plus spécifiques mais qui portent sur une population réduite. Les concepts C126, C117, C127 et C124 sont subsumés par le concept C110, donc toutes les règles extraites de ces concepts sont I-subsumées par la règle R3. Si l'expert veut ajouter la propriété vertébré, il descend au concept C126, et toutes les règles valides issues de ce concept lui sont proposées. Il peut encore réduire sa population et ajouter d'autres propriétés. Les propriétés qu'il peut rajouter sont : denté du concept C158 ou queue du concept C156. En choisissant de rajouter la propriété denté, il obtient deux règles de la même classe R14 : denté, vertébré ⇒ respire et R35 : respire,vertèbre⇒ denté. S'il descent encore dans la hiérarchie vers les concepts C136 et C194, et qu'il trouve que sa population a été trop réduite (le support à trop diminué), il peut s'arrêter à ce niveau de la hiérarchie et ne pas consulter les autres concepts.

Par contre si l'analyste étudie la règle R13 : prédateur ⇒ respire, dans la hiérarchie des règles, il se rendra compte qu'il n'existe aucune propriété pouvant être ajoutée au motif "prédateur, respire" pour avoir une règle valide, car il n'existe aucune descendant dans la hiérarchie.

Cette méthode de classification des règles est simple, et ne demande pas de calcul supplémentaire, car les règles sont directement extraites du treillis. Elle offre à l'analyste une nouvelle méthode pour analyser les règles de façon structurée.

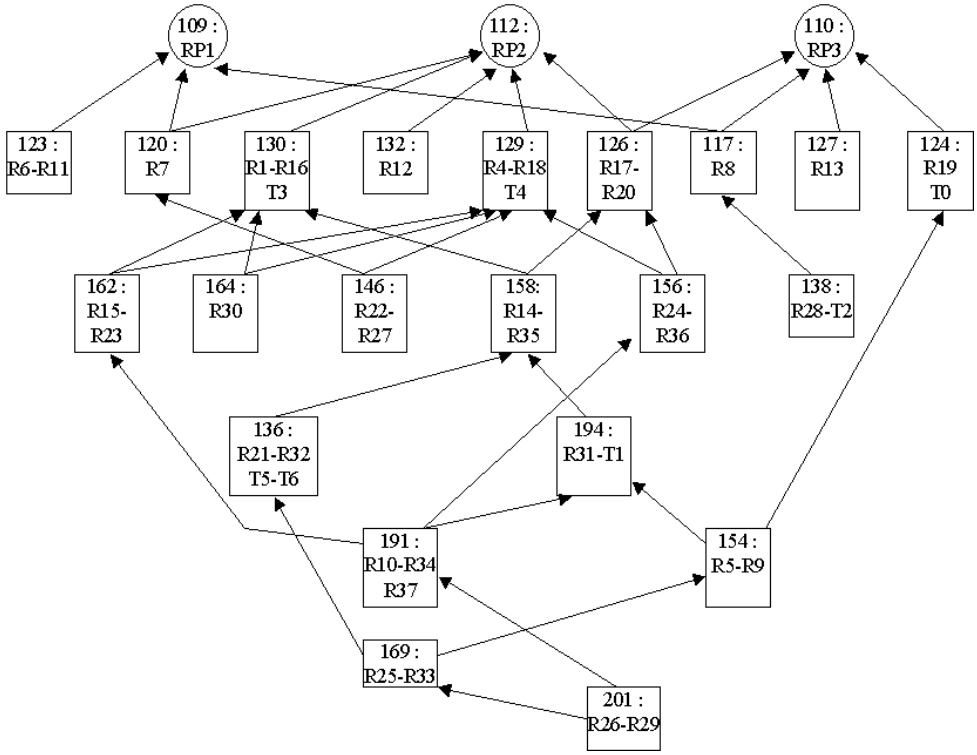

FIG. 3.1 – **Hiérarchie des règles**

3.4 Subsomption des règles avec des propriétés hiérarchisées

Lorsque nous possédons une hiérarchie entre les propriétés, nous définissons la subsomption entre règles d'association que nous nommons la H-subsomption, notée $\sqsubseteq_H$, en nous appuyant sur la définition de subsomption en programmation logique inductive. L'idée de la H-subsomption est de généraliser les règles en s'inspirant de ce qu'a fait Agrawal dans [Agrawal and Srikant, 1995], cependant au lieu de supprimer une règle du type A $\Rightarrow$ B lorsqu'existe la règle Â $\Rightarrow$ B avec Â l'ancêtre de A, nous avons défini la H-subsomption entre ces deux règles et nous les avons gardées toutes les deux.

Avant de définir la H-subsomption entre règles avec des propriétés hiérarchisées, nous nous imposant qu'une propriété ne peut pas être son propre ancêtre pour éviter un cycle dans la hiérarchie. Nous allons éliminer les règles du type A$\Rightarrow$Â et les règles du type Â$\Rightarrow$A car ce sont des règles non informatives.

Définition 19 (H-Subsomption des règles)
Soient deux règles R1 : $A \Rightarrow B$ et R2 : $C \Rightarrow D$, R1 $\sqsubseteq_H$ R2 si et seulement si :

1. *C est ancêtre de A et B = D ($A \Rightarrow B \sqsubseteq_H \hat{A} \Rightarrow B$)*
2. *D est ancêtre de B et A = C ($A \Rightarrow B \sqsubseteq_H A \Rightarrow \widehat{B}$)*
3. *B est ancêtre de A et D est ancêtre de B ($A \Rightarrow B \sqsubseteq_H \hat{A} \Rightarrow \widehat{B}$)*

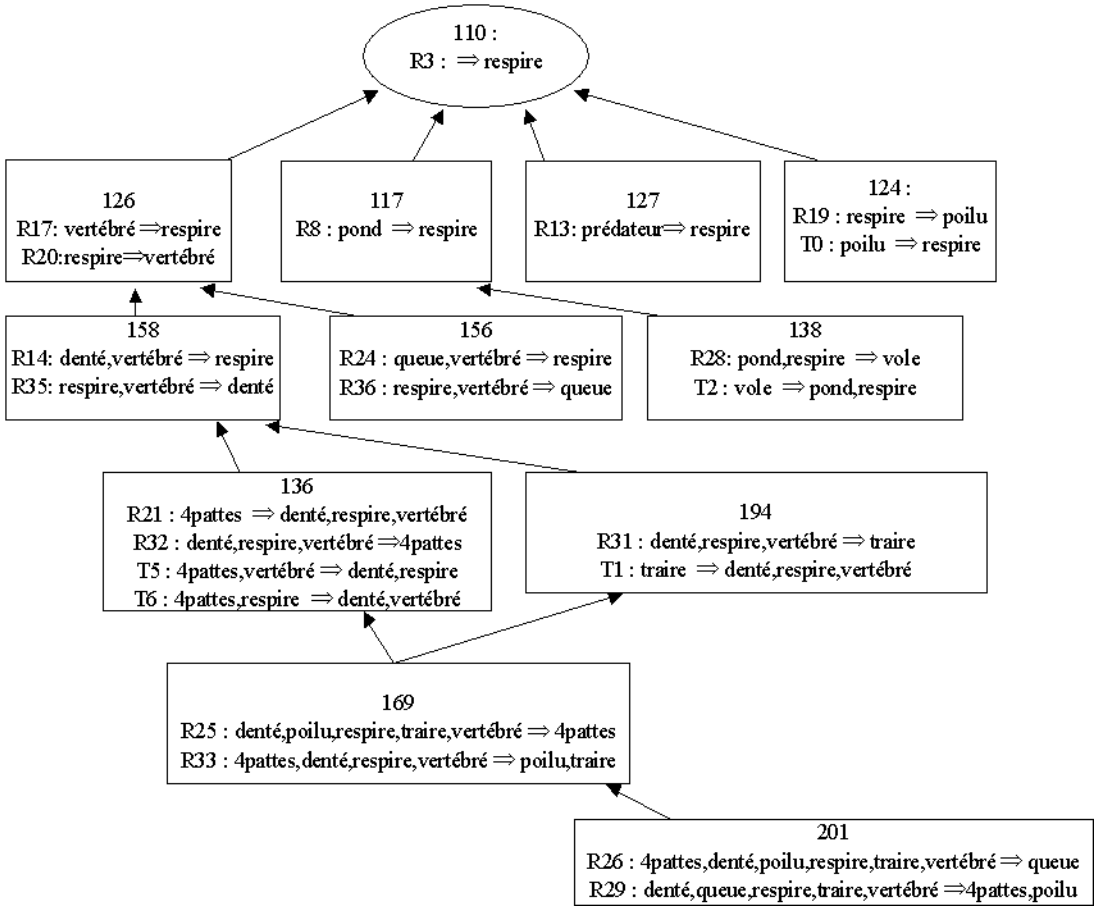

FIG. 3.2 – **Une partie de la hiérarchie des règles**

3.4.1 Interprétation

La figure 3.3 montre la hiérarchie de la règle A ⇒ E.

Nous allons décrire comment cette généralisation a été faite : Il y a deux types de généralisation différentes notées dans la figure 3.3 par 1 et 2 :

1. Généralisation de la partie droite de la règle : ce type de généralisation est directe car nous avons Ê qui est ancêtre de E, ce qui signifie aussi que Ê = E ⊔ X, et d'après les propriétés des règles présentées dans la section 1.4.5 si A ⇒ E alors pour tout ensemble de propriétés X nous pouvons déduire : A ⇒ E ⊔ X et donc A ⇒ Ê. Nous n'avons pas besoin de vérifier que cette règle est valide car le support(Ê ∪ A) ≥ support(E ∪ A) et la confiance(Ê ∪ A) = support (Ê ∪ A) / support (A) est ≥ confiance (E ∪ A) = support (E ∪ A)/support (A).
2. Généralisation de la partie gauche de la règle : Cette partie de généralisation est de nature inductive. Nous considérons la règle A ⇒ Â comme étant une théorie, car c'est une règle totale. À partir de cette théorie et de la règle Â ⇒ E, nous pouvons déduire la règle A ⇒ E, et donc la règle Â ⇒ E subsume la règle A ⇒ E, par rapport à la théorie A ⇒ Â . Pour ce type de généralisation, le support(Â ⇒ E) reste ≥

minsupp mais, par contre, il faut vérifier la confiance($\hat{A} \Rightarrow E$) pour voir si la règle reste valide car le support de la partie gauche de la règle a augmenté.

3.4.2 Propriétés de la H-subsomption

- transitivité : si R1 $\sqsubseteq_H$ R2 et R2 $\sqsubseteq_H$R3 alors : R1 $\sqsubseteq_H$ R3
- réflexivité : R $\sqsubseteq_H$ R
- anti-symétrie : R1 $\sqsubseteq_H$ R2 et R2 $\sqsubseteq_H$ R1 alors R1 = R2.

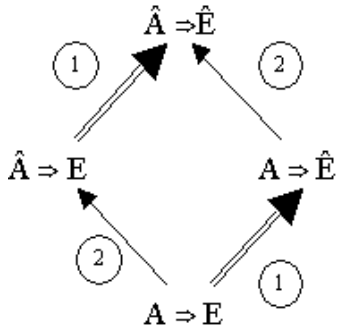

FIG. 3.3 – **Généralisation d'une règle**

3.4.3 Expérimentation sur des règles avec propriétés hiérarchisées

Pour le deuxième exemple, nous avons créé une base de données de 6 individus et de 6 propriétés. Le tableau est présenté dans TAB.3.1.

R	Algèbre	Algorithmique	Probabilité	QoS	PeertoPeer	Biologie
I_1	1	1	1	1	1	0
I_2	1	1	0	0	0	1
I_3	0	0	1	0	0	0
I_4	0	1	1	0	1	0
I_5	1	1	1	1	0	0
I_6	0	0	0	0	1	1

TAB. 3.1 – Représentation en tableau de la relation R

Nous avons extrait les règles d'association avec minconf à 0.5 et minsupp à 0.5, nous en avons obtenu 9 règles partielles et 1 règle totale, qui sont présentées dans TAB.3.2

Puis pour chaque règle qui se compose d'une propriété qui possède un ancêtre dans la hiérarchie de FIG.3.4 nous avons généré une nouvelle règle avec l'ancêtre puis nous avons calculé la confiance pour savoir si c'était des règles valides. Les règles résultantes sont présentées dans le tableau TAB.3.3

Indice	Règles	Support	Confiance
P0	⇒ PeertoPeer	0.5	0.5
P1	⇒ Probabilité	0.66	0.66
P2	⇒ Algorithmique	0.66	0.66
P3	⇒ Algèbre	0.5	0.5
P4	Proba ⇒ Algorithmique	0.5	0.75
P5	Algorithmique ⇒ Probabilité	0.5	0.75
P6	⇒ Probabilité,Algorithmique	0.5	0.5
P7	Algorithmique ⇒ Algèbre	0.5	0.75
P8	⇒ Algorithmique,Algèbre	0.5	0.5
T0	Algèbre ⇒ Algorithmique	0.5	1

TAB. 3.2 – Les règles extraites du tableau de l'exemple 2

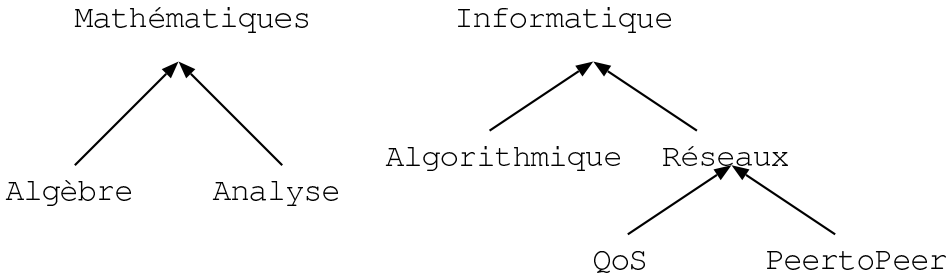

FIG. 3.4 – **Hiérarchie des propriétés**

Indice	Règles	Support	Confiance
R0	⇒ Réseaux	0.66	0.66
R1	⇒ Informatique	0.83	0.83
R2	⇒ Mathématiques	0.83	0.83
R3	Informatique ⇒ Algèbre	0.5	0.6
R4	Algorithmique ⇒ Mathématiques	0.66	0.8
R5	Informatique ⇒ Probabilité	0.5	0.6
R6	Informatique ⇒ Mathématiques	0.66	0.8
R7	⇒ Informatique,Probabilité	0.5	0.5
R8	⇒ Algorithmique,Mathématiques	0.66	0.66
R9	⇒ Informatique,Algèbre	0.5	0.5
R10	⇒ Informatique,Mathématiques	0.66	10.66

TAB. 3.3 – Les règles extraites par généralisation

Pour Les règles P0 et P2 nous avons construit la hiérarchie présentée dans FIG.3.5. À partir de ces deux règles, nous pouvons obtenir les règles ⇒ Réseaux et ⇒ Informatique.

Les règles ⇒ Algèbre et ⇒ Probabilité, sont généralisées par ⇒ Mathématiques et la

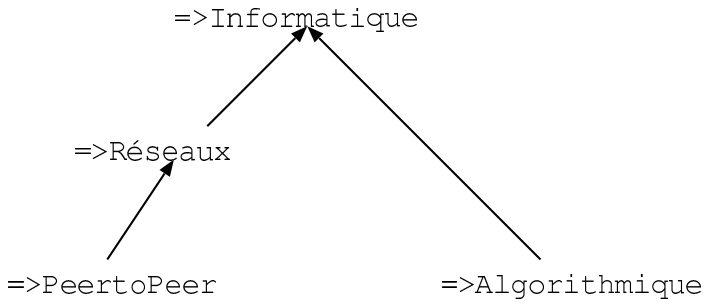

FIG. 3.5 – **Hiérarchisation des règles P0 et P2**

hiérarchie obtenue est présentée dans la figure FIG.3.6.

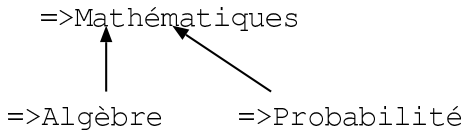

FIG. 3.6 – **Hiérarchisation des règles P1 et P3**

Des règles Algorithmique ⇒ Algèbre et Algorithmique ⇒ Probabilité, la hiérarchie résultante est présentée dans FIG.3.7

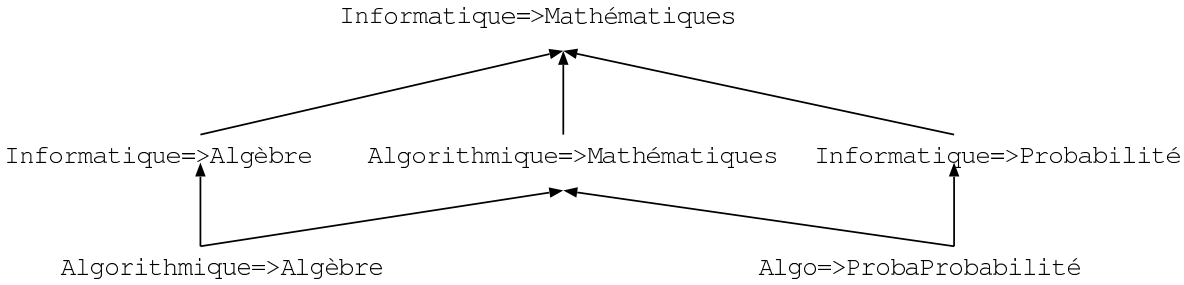

FIG. 3.7 – **Hiérarchisation des règles P5 et P7**

Une autre hiérarchie a été déduit des règles ⇒ Algorithmique, Probabilité et ⇒ Algorithmique, Algèbre dans FIG.3.8.

Supposons qu'un analyste étudie par exemple la règle Informatique ⇒ Algèbre de FIG.3.7 pour savoir quels sont les modules qu'un étudiant en informatique peut poursuivre en même temps dans son cursus. S'il trouve que la propriété informatique est trop générale, il descent dans la hiérarchie pour trouver la règle Algorithme ⇒ Algèbre qui spécialise la partie gauche de la règle. Par contre s'il trouve la règle informatique ⇒ Algèbre trop

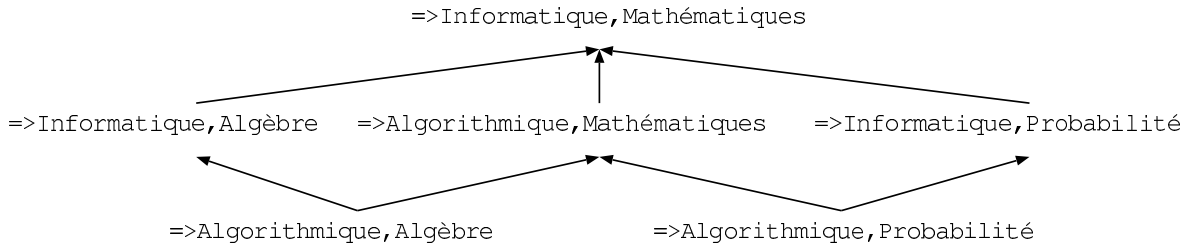

FIG. 3.8 – **Hiérarchisation des règles P6 et P8**

spécifique et qu'il veut généraliser, il monte dans la hiérarchie de FIG.3.7 et il obtient la règle informatique ⇒ Mathématiques. De plus il est sûr qu'elle est valide.

Ainsi, la subsomption entre règles avec des propriétés hiérarchisées permet à l'analyste de ne plus considerer toutes les propriétés des règles au même niveau et d'utiliser cette hiérarchie des propriétés pour généraliser les règles d'association.

Conclusion et perspectives

Lorsque le nombre de règles d'association est important, l'analyste cherche à trouver des liens entre ces règles, pour pouvoir en déduire les connaissances connues dans son domaine, et peut-être même de nouvelles connaissances. Nous avons ajouté la classification des règles d'association dans l'étape de fouille de données pour faciliter le travail de l'analyste lors de l'évaluation et de l'interprétation des règles extraites. Au lieu de lui fournir des règles avec des indices statistiques, nous lui fournissons une hiérarchie de règles d'après les propriétés qui les composent, cette hiérarchie lui permet de faire ressortir les liens donc il a besoin.

Les règles d'association ayant des propriétés non hiérarchisées sont classifiées dès qu'elles sont extraites du treillis de galois. De ce fait cette classification ne demande pas une étape supplémentaire. De plus cette classification offre plusieurs avantages à l'analyste tels que le fait de redéfinir un support minimal s'il trouve que sa population a trop été réduite dans le bas de la hiérarchie, et de voir toutes les règles valides qui sont extraites du même concept (les règles de la même classe).

La structuration des règles d'association avec des propriétés hiérarchisées permet de tenir compte des liens entre les différentes propriétés et de pouvoir généraliser l'une des deux parties de la règle.

La classification des règles d'association dans le cas d'une base de textes sert à relier les textes entres eux. Dans le cas des règles avec des propriétés hiérarchisées, cette relation entre les textes peut être interprétée comme le fait qu'un texte mentionne des termes plus spécifiques qu'un autre, ce qui peut aider l'expert à classifier ces textes.

Comme perspectives, nous envisageons de traiter non plus des propriétés booléennes, mais des propriétés de type multi-valuées, intervalle ou histogramme, et d'adapter la classification des règles dans ce nouveau contexte, ce qui conduit entre autre à adapter les algorithmes de construction de treillis de galois, d'extraction de motifs et de règles d'association.

Annexe A

L'exemple de la subsomption entre règles avec des propriétés non hiérarchisées

A.1 Tableau de la relation de l'exemple 1

Les Propriétés des individus sont : poilu, plumage, pond, traire, vole, aquatique, prédateur, denté, vertèbre, respire, venimeux, nageoire, 0patte, 2pattes, 4pattes, 6pattes, 8pattes, queue, domestique.

A.2 Les règles d'association extraites de l'exemple 1

Les règles partielles :
R0 : $\Rightarrow$ pond (S = 0.6 ; C = 0.6)
R1 : $\Rightarrow$ denté vertèbre (S = 0.55 ; C = 0.55)
R2 : $\Rightarrow$ vertèbre (S = 0.77 ; C = 0.77)
R3 : $\Rightarrow$ respire (S = 0.77 ; C = 0.77)
R4 : $\Rightarrow$ queue vertèbre (S = 0.7 ; C = 0.7)
R5 : poilu respire $\Rightarrow$ denté traire vertèbre (S = 0.37 ; C = 0.88)
R6 : pond $\Rightarrow$ prédateur (S = 0.32 ; C = 0.54)
R7 : pond $\Rightarrow$ vertèbre (S = 0.37 ; C = 0.62)
R8 : pond $\Rightarrow$ respire (S = 0.37R9 : denté respire traire vertèbre $\Rightarrow$ poilu (S = 0.37 ; C = 0.93)
R10 : denté respire traire vertèbre $\Rightarrow$ queue (S = 0.35 ; C = 0.87)
R11 : prédateur $\Rightarrow$ pond (S = 0.32R12 : prédateur $\Rightarrow$ vertèbre (S = 0.35 ; C = 0.73)
R13 : prédateur $\Rightarrow$ respire (S = 0.3 ; C = 0.63)
R14 : denté vertèbre $\Rightarrow$ respire (S = 0.42 ; C = 0.77)
R15 : denté vertèbre $\Rightarrow$ queue (S = 0.47 ; C = 0.86)
R16 : vertèbre $\Rightarrow$ denté (S = 0.55 ; C = 0.7)
R17 : vertèbre $\Rightarrow$ respire (S = 0.65 ; C = 0.83)
R18 : vertèbre $\Rightarrow$ queue (S = 0.7 ; C = 0.9)

R19 : respire ⇒ poilu (S = 0.42 ; C = 0.54)
R20 : respire ⇒ vertèbre (S = 0.65 ; C = 0.83)
R21 : 4pattes ⇒ denté respire vertèbre (S = 0.37 ; C = 0.93)
R22 : queue vertèbre ⇒ pond (S = 0.35 ; C = 0.5)
R23 : queue vertèbre ⇒ denté (S = 0.47 ; C = 0.67)
R24 : queue vertèbre ⇒ respire (S = 0.57 ; C = 0.82)
R25 : denté poilu respire traire vertèbre ⇒ 4pattes (S = 0.35 ; C = 0.93)
R26 : 4pattes denté poilu respire traire vertèbre ⇒ queue (S = 0.32 ; C = 0.92)
R27 : pond vertèbre ⇒ queue (S = 0.35 ; C = 0.93)
R28 : pond respire ⇒ vole (S = 0.3 ; C = 0.8)
R29 : denté queue respire traire vertèbre ⇒ 4pattes poilu (S = 0.32 ; C = 0.92)
R30 : prédateur vertèbre ⇒ queue (S = 0.3 ; C = 0.85)
R31 : denté respire vertèbre ⇒ traire (S = 0.4 ; C = 0.94)
R32 : denté respire vertèbre ⇒ 4pattes (S = 0.37 ; C = 0.88)
R33 : 4pattes denté respire vertèbre ⇒ poilu traire (S = 0.35 ; C = 0.93)
R34 : denté queue vertèbre ⇒ respire traire (S = 0.35 ; C = 0.73)
R35 : respire vertèbre ⇒ denté (S = 0.42 ; C = 0.65)
R36 : respire vertèbre ⇒ queue (S = 0.57 ; C = 0.88)
R37 : queue respire vertèbre ⇒ denté traire (S = 0.35 ; C = 0.6)

Les règles totales :
R0 : poilu ⇒ respire (S = 0.42 ; C = 1.0)
R1 : traire ⇒ denté respire vertèbre (S = 0.4 ; C = 1.0)
R2 : vole ⇒ pond respire (S = 0.3 ; C = 1.0)
R3 : denté ⇒ vertèbre (S = 0.55 ; C = 1.0)
R4 : queue ⇒ vertèbre (S = 0.7 ; C = 1.0)
R5 : 4pattes vertèbre ⇒ denté respire (S = 0.37 ; C = 1.0)
R6 : 4pattes respire ⇒ denté vertèbre (S = 0.37 ; C = 1.0)

R	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19
Antilope	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0	1	0
Ours	1	0	0	1	0	0	1	1	1	1	0	0	0	0	1	0	0	0	0
Sanglier	1	0	0	1	0	0	1	1	1	1	0	0	0	0	1	0	0	1	0
Buffle	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0	1	0
Veau	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0	1	1
Carpe	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	0	0	1	1
Poisson-chat	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0	0	0	1	0
Guépard	1	0	0	1	0	0	1	1	1	1	0	0	0	0	1	0	0	1	0
Poulet	0	1	1	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	1
palourde	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
crabe	0	0	1	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0
langouste	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0
corbeau	0	1	1	0	1	0	1	0	1	1	0	0	0	1	0	0	0	1	0
chevreuil	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0	1	0
chien-de-mer	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0	0	0	1	0
dauphin	0	0	0	1	0	1	1	1	1	1	0	1	1	0	0	0	0	1	0
pigeon	0	1	1	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	1
canard	0	1	1	0	1	1	0	0	1	1	0	0	0	1	0	0	0	1	0
elephant	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0	1	0
flamant	0	1	1	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	0
puce	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
grenouille	0	0	1	0	0	1	1	1	1	1	1	0	0	0	1	0	0	0	0
giraffe	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0	1	0
moustique	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0
chèvre	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0	1	1
gorille	1	0	0	1	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0
mouette	0	1	1	0	1	1	1	0	1	1	0	0	0	1	0	0	0	1	0
aiglefin	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	0	0	1	0
hamster	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0	1	1

TAB. A.1 – Tableau de la relation de l'exemple 1

R	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19
lièvre	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0	1	0
faucon	0	1	1	0	1	0	1	0	1	1	0	0	0	1	0	0	0	1	0
hareng	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0	0	0	1	0
abeille	1	0	1	0	1	0	0	0	0	1	1	0	0	0	0	1	0	0	1
mouche	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0
kiwi	0	1	1	0	0	0	1	0	1	1	0	0	0	1	0	0	0	1	0
coccinelle	0	0	1	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0	0
alouette	0	1	1	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	0
leopard	1	0	0	1	0	0	1	1	1	1	0	0	0	0	1	0	0	1	0
lion	1	0	0	1	0	0	1	1	1	1	0	0	0	0	1	0	0	1	0
homard	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0

TAB. A.2 – Tableau de la relation de l'exemple 1

Bibliographie

[Agrawal et al., 1993] Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large database. In *ACM SIGMOD Conference on Managment of Data*, pages 207–216, Washington.

[Agrawal et al., 1996] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. (1996). Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328, California.

[Agrawal and Srikant, 1995] Agrawal, R. and Srikant, R. (1995). Mining generalized association rules. In *21st VLDB Conference*, Zurich, Swizerland.

[Barbut and Monjardet, 1970] Barbut, M. and Monjardet, B. (1970). *Ordre et classification – Algèbre et combinatoire (2 tomes)*. Hachette, Paris.

[Bastide et al., 2002] Bastide, Y., Taouil, R., Pasquier, N. ans Stumme, G., and Lakhal, L. (2002). Pascal : un algorithme d'extraction des motifs fréquents. In *Technique et science informatiques*, volume 21 - n°1/2002, pages 65–95.

[Cherfi et al., 2003] Cherfi, H., Napoli, A., and Toussaint, Y. (2003). Vers une méthodologie de fouille de textes s'appuyant sur l'extraction de motifs fréquents et de règles d'association. In Gilleron, R., editor, *Actes de la conférence sur l'apprentissage (CAp-03), Laval*, pages 61–76.

[Cherfi and Toussaint, 2002] Cherfi, H. and Toussaint, Y. (2002). Fouille de textes par combinaison de règles d'association et d'indices statistiques. In *CIFT. Volume X-n° X.*

[Cornuéjols and Miclet, 2001] Cornuéjols, A. and Miclet, L. (2001). *Apprentissage artificiel.* EYROLLES, deuxième edition.

[Fayyad et al., 1996] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. In *From Data Mining to Knowledge Discovery*, volume ISBN 0-262-56097-6, page chapitre1.

[Forsyth, 1991] Forsyth, R. (1991). Uci machine learning repository content summary. In *http ://www.ics.uci.edu/ mlearn/MLSummary.html.*

[Ganter and Wille, 1999] Ganter, B. and Wille, R. (1999). *Formal Concept Analysis.* Springer, Berlin.

[Godin et al., 1995] Godin, R., Mineau, R., Missaoui, R., and Mili, H. (1995). Méthodes de classification conceptuelle basées sur les treillis de galois et applications. In *Revue d'intelligence artificielle 9(2)*, pages 105–137.

[Godin and Missaoui, 1994] Godin, R. and Missaoui, R. (1994). An incremental concept formation approach for learning from databases. volume 133, pages 387–419.

[Godin et al., 1991] Godin, R., Missaoui, R., and Alaoui, H. (1991). Learning algorithms using a galois lattice structure. In *Third International Conference on Tools for Artificial Intelligence*, pages 22–29.

[Guénoche, 1990] Guénoche, A. (1990). Construction du treillis de galois d'une relation binaire. In *Revue Math. Inf. Sci. Hum, 109*, pages 41–53.

[Maedche and Staab, 2000a] Maedche, A. and Staab, S. (2000a). Discovering conceptual relation from text. In *Proceeding of the 14th European Conference on artifical intelligence*, pages 321–325, Berlin, Germany.

[Maedche and Staab, 2000b] Maedche, A. and Staab, S. (2000b). Miming ontologies from text. In *Knowledge Acquisition, Modeling and Management*, pages 189–202, France.

[Napoli, 1997] Napoli, A. (Decembre1997). Une introduction aux logiques de description. In *Raport de recherche INRIA n°3314*.

[Polaillon, 1998] Polaillon, G. (1998). *Organisation et interprétation par les treillis de Galois de données de type multivalué, intervalle ou histogramme*. Thèse d'informatique, Université Paris IX-Dauphine.

[Simon, 2000] Simon, A. (2000). *Outils classificatoires par objets pour l'extraction de connaissances dans des bases de données*. Thèse d'informatique, Université Henri Poincaré, Nancy1.

[Toussaint, 2004] Toussaint, Y. (2004). Extraction de connaissances à partir de textes structurés. In *Document Numérique*, volume 8, pages 1–24.

[Valtchev et al., 2003] Valtchev, P., Godin, R., Missaoui, R., Grosser, D., Roume, C., and Rouane-Hacene, A. (2003). Galicia. In *http ://www.iro.umontreal.ca/ galicia/*, Montréal, Québec.

[Wille, 1992] Wille, R. (1992). Concept lattices and conceptual knowledge systems. In *Computers and Mathematics with Applications*, volume 23(6-9), pages 493–515.