Treillis de Galois Alpha

Véronique Ventos, Henry Soldano, Thibaut Lamadon

Treillis de Galois Alpha

Véronique Ventos[1], Henry Soldano[2], Thibaut Lamadon[1]

[1] LRI, UMR-CNRS 8623, Université Paris-Sud, 91405 Orsay, France
ventos@lri.fr

[2] L.I.P.N, UMR-CNRS 7030, Université Paris-Nord,
93430 Villetaneuse, France
soldano@lipn.univ-paris13.fr

Résumé : Dans beaucoup d'applications il est utile de représenter une grande quantité de données en les regroupant en une hiérarchie de classes décrites à un niveau d'abstraction adéquat. La représentation générique que nous utilisons ici est un treillis de Galois, i.e. un treillis correspondant au partitionnement des termes d'un langage de représentation en classes d'équivalence relativement à leur *extension* (l'extension d'un terme est la partie d'un ensemble d'instances qui satisfait ce terme). L'avantage d'une telle structure est qu'elle représente exhaustivement l'ensemble des concepts qui peuvent être distingués en fonction de leur extension. Son principal désavantage est sa taille qui peut être très grande dans le cas d'applications réelles. Nous proposons ici de réduire la taille du treillis, simplifiant ainsi la représentation des données, tout en conservant sa structure formelle de treillis de Galois et son exhaustivité. Pour cela nous utilisons une partition préliminaire des données correspondant à l'association d'un type à chaque instance. En redéfinissant la notion d'extension d'un terme de manière à tenir compte, à un certain degré α, de cette partition, nous aboutissons à des treillis de Galois particuliers appelés *treillis de Galois Alpha*.

Mots-clés : Regroupement conceptuel,Treillis de Galois

1 Introduction

Dans beaucoup d'applications il devient primordial d'aider interactivement un utilisateur à accèder à une grande quantité de données. Une manière de procéder consiste à regrouper les instances en classes, elle-mêmes organisées en une hiérarchie, et décrites à un niveau d'abstraction adéquat. La représentation de départ que nous utilisons ici est un treillis de concepts. Dans un tel treillis chaque noeud correspond à une classe représentée par son *extension* (les instances de la classe) et son *intention* (les propriétés communes à la classe exprimées comme un terme d'un langage de classes). Le treillis de concepts constitue une représentation exhaustive des concepts sous-jacents à l'ensemble des instances : tout sous-ensemble des instances constituant l'extension d'un terme du langage est associé à un et un seul noeud du treillis. Son principal désavantage

est sa taille qui peut être tres grande dans le cas d'applications réelles. Plusieurs techniques ont été proposées pour réduire la taille du treillis en éliminant une partie de ses noeuds (e.g. (Hereth *et al.*, 2000)). En particulier un treillis de concepts *fréquents* (Waiyamai & Lakhal, 2000) représente la partie supérieure d'un treillis de concepts : seuls les noeuds dont l'extension est suffisament grande (relativement à un seuil) sont représentés. Dans l'approche présentée ici nous contrôlons le nombre de noeuds du treillis en tenant compte, dans une certaine mesure associée à un degré α, d'une partition *a priori* des données. Cette partition est constituée d'un ensemble de *classes de base* : chaque classe est associée à un *type de base* : celui des instances qui la constituent. Ainsi dans un jeu de données concernant le catalogue électronique de produits informatiques C/Net (http ://www.cnet.com), il y a 59 types de base différents (e.g. *Laptops*, *Harddrive*, *NetworkStorage*) pour 2274 instances. Les classes de base sont alors utilisées pour ajouter un critère de fréquence *locale* à la notion d'$extension$ de la manière suivante : une instance i appartient à $ext_\alpha(T)$ (la α-$extension$ d'un terme T du langage de classes), lorsque, d 'une part, elle appartient à l'extension de T, $ext(T)$, (i.e. i a toutes les propriétés qu'exprime T), et d'autre part, au moins α % des instances de la classe de base de i appartiennent aussi à $ext(T)$. Cette nouvelle notion d'α-$extension$ induit de nouveaux treillis de concepts, plus flexibles, qui ont formellement la structure de treillis de Galois, et que nous appelons ici *treillis de Galois Alpha* ou plus simplement *treillis Alpha*. Pour en revenir à l'exemple du catalogue electronique, considérons un treillis de concepts fréquents. La propriété "support" n'apparaît dans aucun noeud car elle n'est pas globalement fréquente (13 produits sur 2274). Dans un treillis Alpha (avec α plus petit que 92) la propriété "support" apparaît dans au moins un terme, car dans la classe de base $HardDrives$ 92 % des instances sont vendues avec un "support". Autrement dit le treillis de Galois Alpha introduit une notion de fréquence *locale* qui permet de faire apparaître dans le treillis des propriétés qui ne sont fréquentes que dans certaines classes de base.

Nous montrons que pour un même langage et un même ensemble d'instances, le *treillis de Galois Alpha* est plus *grossier* que le treillis de concepts : les noeuds du *treillis de Galois Alpha* constituent un sous-ensemble des noeuds du treillis de concepts. Nous montrons également que les valeurs de α définissent un ordre total sur les *treillis de Galois Alpha* : le *treillis de Galois Alpha* induit par $ext_{\alpha 1}$ est plus grossier que celui induit par $ext_{\alpha 2}$ si $\alpha 1 \geq \alpha 2$. Cet ordre total permet en particulier de faire des "zooms" successifs sur les *treillis de Galois Alpha* permettant ainsi de contrôler la taille du treillis et de tenir compte à la fois des limitations de l'utilisateur (ce qu'il peut appréhender) et de ses désirs (ce qui l'interesse). L'idée est de construire dans un premier temps un treillis grossier (avec ext_{100}) puis de raffiner une partie de ce treillis, délimitée par deux noeuds (ascendant/descendant), en faisant décroître la valeur de α. Une telle stratégie de zoom/raffinement successifs est implémentée dans le système ZooM (Pernelle *et al.*, 2002) dédié à la réprésentation de données par des treillis à différents niveaux d'abstraction.

Le cadre général des treillis de Galois est donné en section 2. En section 3 nous présentons, et illustrons sur un exemple simple, les *treillis de Galois Alpha*. La section 4 presente des expériences, menées sur les données de C/net, qui mettent l'accent sur la robustesse de cette représentation en particulier en présence de données *exceptionnelles* re-

lativement à leur classe de base (avec des valeurs de α proches de 0 ou de 100). La section 5 traite d'abord des règles d'implications particulières associées aux *treillis de Galois Alpha*, puis de la comparaison du point de vue ensembliste associé à la notion d' α*-extension* avec celui de la théorie des ensembles "rugueux" (rough sets). Enfin la conclusion relie ce travail à d'autres travaux sur les treillis de concepts et trace quelques perspectives de recherche.

2 Préliminaires et définitions

Les principales définitions, preuves et résultats concernant les correspondances et treillis de Galois sont présentées entre autres dans (Birkhoff, 1973). D'autres résultats sur les treillis de Galois redéfinis dans le champ de l'Analyse Formelle de Concepts apparaissent dans (Ganter & Wille, 1999) et, dans le cadre de l'Analyse des Objets Symboliques, dans (Bock & Diday, 2000). Nous utilisons ici une présentation plus large que celle de (Ganter & Wille, 1999) dans la mesure où nous ferons varier par la suite la notion d'*extension*. Dans la suite du papier nous appelons treillis de Galois la structure formelle que nous définissons ci-dessous et réservons le terme *treillis de concepts* aux treillis de Galois présentés dans (Ganter & Wille, 1999).

Définition 1 (Ensembles ordonnés et treillis)
Un ensemble ordonné est un couple (M,$\leq$) où M est un ensemble et $\leq$ une relation d'ordre (reflexive, antisymétrique et transitive) sur M. Un ensemble ordonnné (M,$\leq$) est un treillis ssi pour tout couple d'éléments (x,y) de M il existe un plus petit majorant (ou supremum) $x \vee y$, et un plus grand minorant (ou infimum) $x \wedge y$

Définition 2 (Correspondance de Galois)
Soient m1 : P $\rightarrow$ Q et m2 : Q $\rightarrow$ P des fonctions définies sur deux ensembles ordonnés (P,$\leq_P$) et (Q,$\leq_Q$). (m1,m2) est une correspondance de Galois si pour tout p, p1, p2 de P et pour tout q, q1, q2 de Q :
C1- p1 $\leq_P$ p2 $\Rightarrow$ m1(p2) $\leq_Q$ m1(p1)
C2- q1 $\leq_Q$ q2 $\Rightarrow$ m2(q2) $\leq_P$ m2(q1)
C3- p $\leq_P$ m2(m1(p)) et q $\leq_Q$ m1(m2(q))

L'exemple ci-dessous sera utilisé pour illustrer les différentes notions presentées en section 2 et 3.

Exemple 1
Les deux ensembles ordonnés sont ($\mathcal{L}$,$\preceq$) et ($\mathcal{P}(I)$, $\subseteq$) :
- $\mathcal{L}$ est un langage dont un terme est un sous-ensemble d'un ensemble d'attribut $\mathcal{A}$ = {t1, t2, t3,a3,a4,a5,a6,a7,a8}. Ici c1 $\preceq$ c2 signifie que le terme c1 est moins spécifique que le terme c2 (par exemple, {a3,a4} $\preceq$ {a3,a4,a6}),
- I est un ensemble d'individus = {i1,i2,i3,i4,i5, i6,i7,i8}.
Soient int et ext deux fonctions telles que :
int : $\mathcal{P}(I) \rightarrow \mathcal{L}$ et ext : $\mathcal{L} \rightarrow \mathcal{P}(I)$ avec :
$int(e1)$ = ensemble des attributs communs à tous les individus de $e1$

$ext(c1)$ = $\{i \in I$ *tels que* $i\ isa\ c1\}$ *où* isa *est l'appartenance usuelle d'un individu à un terme.* $ext(c1)$ *est donc l'ensemble des individus qui ont tous les attributs de* $c1$.
L'exemple 1 est décrit en Figure 1 : chaque ligne i *représente l' intention* $int(\{i\})$ *d'un individu de* I *et chaque colonne* j *représente l' extension* $ext(\{j\})$ *d'un attribut de* $\mathcal{A}$.
int *et* ext *forment une correspondance de Galois sur* $\mathcal{L}$ *et* $\mathcal{P}(I)$,

	t1	t2	t3	a3	a4	a5	a6	a7	a8
i1	1			1	1		1		1
i2	1			1		1	1		
i3		1			1		1		1
i4		1			1		1	1	
i5		1		1			1		1
i6			1	1			1		1
i7			1	1			1		1
i8			1	1		1	1		1

FIG. 1 – Exemple 1. $Tab(i, j)$ = 1 si le j^{eme} attribut appartient au i^{eme} individu.

Nous rappelons ci-dessous la définition d'un opérateur de *fermeture*

Définition 3 (Fermeture)
w est un opérateur de fermeture sur un ensemble ordonné (M,$\leq$*) ssi pour tout couple (x,y) d'éléments de M on a :*
- $x \leq w(x)$ *(extensivité)*
- Si $x \leq y$ *alors* $w(x) \leq w(y)$ *(monotonie)*
- $w(x) = w(w(x))$ *(idempotence)*
Un élement de M tel que x=w(x) est appelé un terme fermé de M relativement à w.

Dans une correspondance de Galois, $m1 \circ m2$ et $m2 \circ m1$ sont des opérateurs de fermeture pour P et Q.
Example : *Dans l'exemple 1,* $ext(\{a4\}) = \{i1, i3, i4\}$, $int(\{i1, i3, i4\}) = \{a4, a6\}$.
Le terme $\{a4, a6\}$ *est fermé car* $int(ext(\{a4\}) = \{a4, a6\}$.

Définition 4 (Treillis de Galois)
Soient m1 : $P \rightarrow Q$ *et m2 :* $Q \rightarrow P$ *deux fonctions définies sur les treillis (P,*$\leq_P$*) et (Q,*$\leq_Q$*), telles que (m1,m2) est une correspondance de Galois.*
Soit G=$\{$ *(p,q), où p est un élément de P et q un élément de Q, tels que p=m2(q) et q = m1(p)*$\}$
Soit $\leq$ *la relation définie par : (p1,q1)* $\leq$ *(p2,q2) ssi q1* $\leq_Q$ *q2.*
(G,$\leq$*) est un treillis appelé treillis de Galois. Nous utiliserons si nécessaire la notation complète* $G(P, m1, Q, m2)$.
Exemple *: Dans l'exemple 1, G=*$\{$*(c,e) où* c *appartient à* $\mathcal{L}$*, et* e *appartient à* $\mathcal{P}(\mathcal{I})$

et tels que e=ext(c) et c=int(e)}. (G, $\leq$) est un treillis de Galois avec $\leq$ définie par : (c,e) $\leq$ (c',e') ssi e $\subseteq$ e' (ce qui est en fait équivalent à c $\succeq$ c'). Le treillis de Galois correspondant à l'exemple 1 est presenté Figure 2.

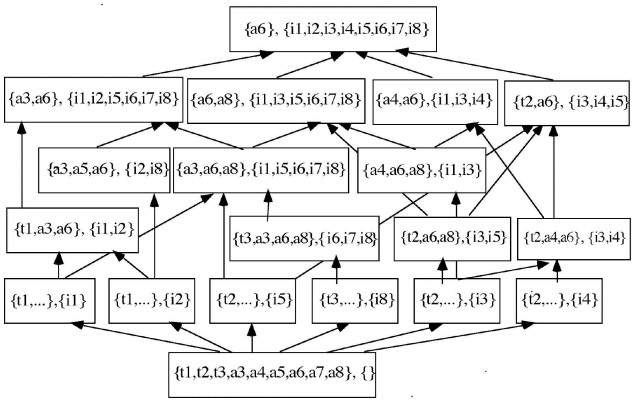

FIG. 2 – Le treillis de Galois correspondant à l'exemple 1.

Remarquons qu'un noeud d'un treillis de Galois est un couple d'éléments fermés de P et Q. De plus les fonctions $m1$ et $m2$ définissent des relations d'équivalence sur les treillis P et Q :

Définition 5 (Relations d'équivalence sur P et Q)
Soient $\equiv_P$ et $\equiv_Q$ les relations d' équivalence définies sur P et sur Q par $m1$ et $m2$, c.à.d. soient p1 et p2 des éléments de P et q1, q2 des élements de Q, alors :
$p1 \equiv_P p2$ *ssi* $m1(p1) = m1(p2)$, *et* $q1 \equiv_Q q2$ *ssi* $m2(q1) = m2(q2)$

Lemme 1
Soient p un élément de P, et q un élément de Q, m2(m1(p)) est l'unique plus grand élément de la classe d' équivalence de $\equiv_P$ contenant p et m1(m2(q)) est l'unique plus grand élément de la classe d' équivalence de $\equiv_Q$ contenant q.

Ainsi, une propriété caractéristique des treillis de Galois est que chaque noeud (p, q) est constitué de représentants de classes d'équivalence de $\equiv_P$ et de $\equiv_Q$.

Dans notre exemple, nous avons utilisé le langage $\mathcal{L}$, défini comme l'ensemble $\mathcal{P}(\mathcal{A})$ des parties d'un ensemble d'attributs A, comme premier treillis, et l'ensemble $\mathcal{P}(\mathcal{I})$ des parties d'un ensemble d'individus I comme second treillis. Ces treillis, associés aux deux fonctions int et ext, définissent un treillis de Galois particulier nommé *treillis de concepts* (Ganter & Wille, 1999). Dans un treillis de concepts, un noeud (c, e) est un concept, c est l' *intention* et e est l'*extension* du concept. Les treillis de concepts sont intéressant à la fois d'un point de vue pratique, dans la mesure où ils expriment d'une manière rigoureuse les deux facettes d'un concept, et d'un point de vue théorique car il a été montré que tout treillis fini peut se réecrire comme un treillis de concepts (Ganter & Wille, 1999).

3 Treillis de Galois Alpha

Dans ce qui suit nous considérons, sans perte de généralité, $\mathcal{L} = \mathcal{P}(A)$ et prenons comme point de départ le treillis de concepts $G(\mathcal{L}, ext, \mathcal{P}(I), int)$ pris comme exemple ci-dessus. Puis nous présentons une variante de ext dont la relation d'équivalence associée $\equiv^{.}_{\mathcal{L}}$ est plus grossière que celle associée à ext (*cf* la définition 11) ce qui conduit à des classes d'équivalence plus grandes sur $\mathcal{L}$ et donc à un treillis de Galois constitué d'une partie seulement des noeuds du treillis de concepts associé à ext.

La nouvelle notion d'extension repose sur l'association d'un type pré-défini à chaque individu. Les individus sont regroupés en *classes de base* correspondant à ces types. La première idée consiste alors à considérer, pour construire le treillis de concepts, les *classes de base* plutôt que les individus (cf (Pernelle *et al.*, 2001)).
Supposons par exemple que les attributs $t1, t2, t3$ correspondent aux trois types possibles pour les individus de l'ensemble I de l'exemple 1. A partir de ces types on engendre 3 classes de base $BC1$, $BC2$, $BC3$ dont les descriptions sont les suivantes :
BC1={i1,i2}, int(BC1)= {t1,a3,a6} (les attributs communs à $i1$ et $i2$) ; BC2={i3,i4,i5}, int(BC2)= {t2,a6} ; BC3={i6,i7,i8}, int(BC3)= {t3,a3,a6,a8}.
Il est maintenant possible de construire un treillis de concepts avec un nouvel ensemble de trois individus : {bc1,bc2,bc3} (appelons les les *prototypes* de leur classe de base respective) tels que, pour tout index i, $int(BCi) = int(\{bci\})$. Ce treillis de Galois, représenté Figure 3, est un cas particulier de treillis de Galois Alpha, beaucoup plus petit que le treillis de concepts original.
Cependant il semble intéressant de concevoir une approche intermédiaire, dans laquelle on conserverait une influence de la partition des données en classes de base sans pour autant se restreindre, comme ci-dessus, à ne considérer que des réunions de classes de base dans les extensions.

C'est cette approche intermédiaire qui conduit aux *treillis de Galois Alpha*.

3.1 Alpha définitions

Définition 6 (Alpha satisfaction)
Soit α un nombre entre [0,100]. Soient e = $\{i_1, \ldots, i_n\}$ un ensemble d'individus, et T un terme de $\mathcal{L}$.

$$e\ \alpha - satisfait\ T\ (e\ sat_\alpha\ T)\ ssi\ \mid ext(T) \cap e \mid \geq \frac{\mid e \mid .\alpha}{100}$$

La α-satisfaction d'un terme de $\mathcal{L}$ étant définie relativement à un ensemble d'individus, nous l'utilisons pour tester si un pourcentage α % des instances d'une classe de base satisfait un terme de $\mathcal{L}$. Nous ajoutons alors cette contrainte à la relation d'appartenance isa entre les individus et les termes de $\mathcal{L}$. Nous appelons cette nouvelle notion (appartenance de l'individu à un terme plus α-satisfaction de la classe de base de l'individu à ce même terme) la α-*appartenance*.

Définition 7 (Alpha appartenance)
Soit I un ensemble d'individus et $\mathcal{BC}$ une partition finie de I (en classes de base). Soit BCl : $I \rightarrow \mathcal{BC}$ la fonction telle que $BCl(i)$ est la classe de base de l' individu i, et soit

T un terme de $\mathcal{L}$, alors :

$$i\ isa_\alpha\ T\ ssi\ i\ isa\ T\ et\ BCl(i)\ sat_\alpha\ T$$

Exemple (exemple 1) *: Soit T=$\{a6,a8\}$, ext(T) = $\{i1,i3,i5,i6,i7,i8\}$.*
$BC1\ sat_{50}\ T$ puisque $i1\ isa\ T$ et $\mid BC1 \mid = 2$. En conséquence $i1\ isa_{50}\ T$. $BC2\ sat_{60}\ T$ puisque $\mid ext(T) \cap BC2 \mid \geq \frac{|BC2|.60}{100}$. Ainsi nous avons $i3$ et $i5\ isa_{60}\ T$. Finalement $BC3\ sat_{100}\ T$ puisque 100 % des individus de $BC3$ appartiennent à l'extension de T. Ainsi nous avons $i6$, $i7$, et $i8\ isa_{100}\ T$.

Nous utilisons maintenant la *Alpha appartenance* pour définir la notion d'extension que nous utiliserons dans les treillis de Galois Alpha.

Définition 8 (Alpha extension d'un terme)
La α-extension d'un terme T dans I, relativement à la partition $\mathcal{BC}$, est définie comme suit :

$$ext_\alpha(T) = \{i \in I \mid i\ isa_\alpha\ T\}$$

Exemple (exemple 1) *: Soit T=$\{a6,a8\}$, ext_0(T)= ext(T) = $\{i1, i3,i5, i6,i7,i8\}$*
ext_{60}(T)= $\{i3,i5,i6,i7,i8\}$ et ext_{100}(T)= $\{i6,i7,i8\}$

La proposition suivante définit une nouvelle correspondance de Galois à partir des fonctions int et ext_α. Elle nécessite de définir un treillis E_α de sous-ensembles de I correspondant à une partie seulement de $\mathcal{P}(\mathcal{I})$. Un élément de E_α est constitué de parties suffisamment grandes de différentes classes de base (c.à.d de parties dont la cardinalité est supérieure ou égale à α % de la classe de base correspondante).

Proposition 1
Soit E_α le sous-ensemble de $\mathcal{P}(\mathcal{I})$ définit comme suit : :
$E_\alpha = \{e \in \mathcal{P}(I) \mid \forall i \in e, \mid e \cap BCl(i) \mid \geq \frac{|BCl(i)|.\alpha}{100}\}$.
Alors : int et ext_α définissent une correspondance de Galois entre $\mathcal{L}$ et E_α.
Preuve *: La preuve de cette proposition repose sur le 1) du théoreme 1 présenté à la section suivante et est en conséquence donnée après l'énoncé de ce théorème, sous la forme de la preuve d'un corollaire.*

Nous pouvons maintenant définir les *treillis de Galois Alpha* associés à cette nouvelle correspondance de Galois.

Définition 9 (Treillis de Galois Alpha)
Soit G=$\{$ (c,e) où c est un élément de $\mathcal{L}$, e est un élément de E_α, e=ext_α(c) et c= int(e)$\}$. Le treillis de Galois correspondant $G(\mathcal{L}, ext_\alpha, E_\alpha, int)$ est appelé treillis de Galois Alpha et est noté G_α.

Lorsque α = 0, on a E_α = $\mathcal{P}(I)$ et ext_α = ext. Ainsi le treillis de Galois Alpha est dans ce cas le treillis de concepts associé aux mêmes attributs et aux mêmes instances. Lorsque α = 100, l'extension associée à un noeud de ce treillis de Galois Alpha est constitué de classes de base entières . En conséquence le treillis de Galois Alpha est le

treillis de concepts obtenu en considérant comme instances les *prototypes* des classes de base.

Le treillis de Galois Alpha G_{100} de l'exemple 1 est représenté Figure 3. La Figure 4 présente la partie supérieure de G_{60}. Notons que les *intentions* des noeuds de G_{100} sont aussi les *intentions* de noeuds de G_{60} qui sont eux mêmes les intentions de noeuds du treillis de concepts initial G_0 (voir Figure 2).

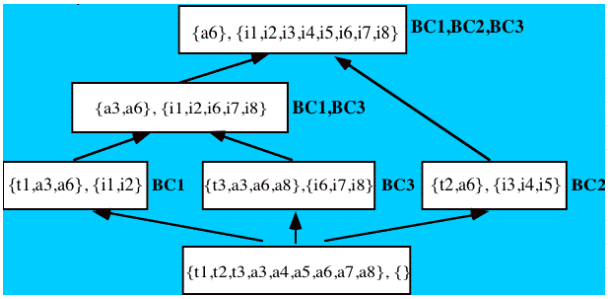

FIG. 3 – α =100 : le treillis de Galois Alpha G_{100} de l'exemple 1 est beaucoup plus petit que le treillis de concepts initial présenté Figure 2.

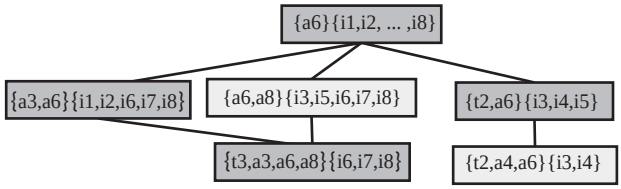

FIG. 4 – α = 60 : La partie supérieure de G_{60} de l'exemple 1. Les nouveaux noeuds, relativement à G_{100} sont d'un gris plus clair.

Dans la section suivante nous détaillons comment s'ordonnent les treillis Alpha.

3.2 Ordres sur les treillis de Galois Alpha

Dans (Ganter & Kuznetsov, 2001) les auteurs formalisent l'extension de l'analyse formelle de concepts à des langages plus sophistiqués. Une notion de *projection* est utilisée pour réduire le langage et ainsi réduire la taille du treillis de Galois. Indépendamment, (Pernelle *et al.*, 2002) utilise la même notion de projection pour faire varier le langage mais l'utilise également en la nommant projection *extensionelle*, pour modifier la fonction d'extension *ext*.

Nous rappelons ci-dessous la notion générale de projection, ainsi qu'une relation d'ordre sur les relations d'équivalence :

Définition 10 (Projection)
Proj est une projection d'un ensemble ordonné (M,≤) ssi pour tout couple (x,y) d' éléments de M :

x ≥ Proj(x) (minimalité)
if x ≤ y then Proj(x) ≤ Proj(y) (monotonie)
Proj(x) = Proj(Proj(x)) (idempotence)

En changeant *ext*, une projection extensionnelle transforme un treillis de Galois en un treillis plus petit pour lequel la relation d' équivalence $\equiv'_{\mathcal{L}}$ est plus grossière et dont les classes d' équivalence sur $\mathcal{L}$ incluent celles de la relation d'origine :

Définition 11
Soit $\equiv^1_{\mathcal{L}}$ la relation associée à la fonction ext1 et $\equiv^2_{\mathcal{L}}$ celle associée à ext2, alors, $\equiv^2_{\mathcal{L}}$ est dite plus grossière que $\equiv^1_{\mathcal{L}}$ ssi pour tout couple (c1,c2) d' éléments de $_{\mathcal{L}}$ on a :
si ext1(c1) = ext1(c2) alors ext2(c1) = ext2(c2).

Le théorème suivant est prouvé dans (Pernelle *et al.*, 2002) :

Théorème 1 (Un ordre extensionnel sur les correspondances de Galois)
Considérons les fonctions int et ext définissant une correspondance de Galois sur $\mathcal{L}$ et E et $proj$ une projection de E. Notons E' = proj(E) et ext' = $proj \circ ext$. Alors :
1) int, ext' définissent une correspondance de Galois sur $\mathcal{L}$ et E'.
2) Le treillis de Galois $G'(\mathcal{L}, ext', E', int)$ a la propriété suivante : pour tout noeud g'=(c,e') de G' il existe un noeud g=(c,e) de $G(\mathcal{L}, ext, E, int)$, avec la même $intention$ c, tel que e'=proj(e).
3) $\equiv'_{\mathcal{L}}$ est plus grossière que $\equiv_{\mathcal{L}}$.
Nous dirons alors que G' est plus grossier extensionnellement que G.

Le corollaire suivant de ce théorème démontre la proposition 1.

Corollaire 1
Soit $G(\mathcal{L}, ext$,P(I), $int)$ un treillis de Galois. Soit ext_α=$proj_\alpha \circ ext$ et E_α = $proj_\alpha(\mathcal{P}(\mathcal{I}))$ avec pour tout e de $\mathcal{P}(\mathcal{I})$:

$$proj_\alpha(e) = e - \{i/\ i\ \in e\ et \mid e \cap BCl(i) \mid < \frac{\mid BCl(i) \mid .\alpha}{100}\}$$

1) int, ext_α définissent une correspondance de Galois sur $\mathcal{L}$ et E_α.
2) Le treillis de Galois $G'(\mathcal{L}$, ext_α, E_α, int) est tel que : pour tout noeud g'=(c,e') de G' il existe un noeud g=(c,e) de $G(\mathcal{L}, ext$,P(I), $int)$, avec la même $intention$ c, tel que e'=$proj_\alpha(e)$
preuve : *Le corollaire découle directement du théorème dans la mesure où on prouve que $proj_\alpha$ est une projection (on démontre ainsi la proposition 1).*
$proj_\alpha$(e) est inclus dans e puisqu' on retire des éléments de e, donc $proj_\alpha$ est minimal. Si e est inclus dans e', chaque élément de e retiré par la projection sera aussi retiré de e', $proj_\alpha$(e) est donc inclus dans $proj_\alpha$(e') et $proj_\alpha$ est monotone. Finalement, $proj_\alpha$ est idempotent puisque plus aucun élément de $proj_\alpha$(e) ne peut être retiré par une seconde application de $proj_\alpha$.

De plus nous pouvons ordonner ces fonctions d'extensions selon la valeur de α : Pour tout couple ($\alpha 1$, $\alpha 2$) tel que $\alpha 1 \leq \alpha 2$, on a $ext_{\alpha 2} = proj_{\alpha} \circ ext_{\alpha 1}$ avec $\alpha = \alpha 2$. En consequence, la valeur de α définit un ordre total sur les treillis de Galois Alpha :

Proposition 2 (Un ordre total sur les treillis de Galois Alpha)
Notons $\equiv_{\alpha}$ la relation d' équivalence sur $\mathcal{L}$ associée à ext_{α}. Alors pour tout couple ($\alpha 1$, $\alpha 2$) tel que $\alpha 1 \leq \alpha 2$, $\equiv_{\mathcal{L}}^{\alpha 2}$ est plus grossière que $\equiv_{\mathcal{L}}^{\alpha 1}$
Preuve : *$proj_{\alpha}$ est une projection pour toute valeur de α appartenant à [0,100]. $ext_{\alpha 1}$=$proj_{\alpha} \circ ext$ avec $\alpha = \alpha 1$ et $ext_{\alpha 2}$=$proj_{\alpha} \circ ext_{\alpha 1}$ avec $\alpha = \alpha 2$. Selon le 3) du théorème 1, $\equiv_{\mathcal{L}}^{\alpha 2}$ est alors plus grossière que $\equiv_{\mathcal{L}}^{\alpha 1}$*

Exemple : *$\equiv_{\mathcal{L}}^{100}$ est plus grossière que $\equiv_{\mathcal{L}}^{60}$ qui est elle-même plus grossière que $\equiv_{\mathcal{L}}^{0}$ qui est la relation d'équivalence $\equiv_{\mathcal{L}}$ du treillis de concepts .*

Cette dernière proposition permet de faire des raffinements successifs de treillis de Galois Alpha (voir la section 4).

Il existe aussi un ordre partiel associé à la partition initiale $\mathcal{BC}$ de I en classes de base. Supposons que nous retirions certaines classes de base de I, réduisant ainsi l'ensemble d'instances à I' associé à une partition réduite $\mathcal{BC}'$ constituée des classes de base restantes. Il est aisé de montrer (la preuve est ici omise) qu'il y a une projection $proj$ telle que le treillis E'_{α} correspondant s'écrit simplement $proj(E_{\alpha})$. En conséquence nous avons la propriété suivante :

Proposition 3 (Un ordre partiel sur les treillis de Galois Alpha)
Soit $\mathcal{BC}'$ un sous-ensemble des classes constituant $\mathcal{BC}$, alors, pour la même valeur α, le treillis de Galois Alpha G' associé à $\mathcal{BC}'$ est plus grossier que le treillis de Galois Alpha G associé à $\mathcal{BC}$.

Un cas particulier intéressant est celui de la partition $\{I\}$ constituée d'une seule classe, c'est à dire le cas où toutes les instances partagent le même type. Le treillis de Galois Alpha correspondant est la partie supérieure du treillis de concepts défini par le même langage $\mathcal{L}$ et le même ensemble I d'individus, et est constitué des noeuds dont l'extension est plus grande que $\frac{\alpha}{100}|I|$. Cette structure a été récemment étudiée et a été nommée *treillis Iceberg* ou *treillis de concepts fréquents* (Stumme *et al.*, 2002; Waiyamai & Lakhal, 2000). Dans ces structures $\frac{\alpha}{100}$ correspond à la valeur du seuil sur le support *minsupp*.

Notons que du fait de la proposition 3, le treillis fréquent de chaque classe de base BCi d'une partition $\mathcal{BC}$ est toujours plus grossier que le treillis de Galois Alpha correspondant à $\mathcal{BC}$.

4 Implémentation et Expériences

Le programme ALPHA qui construit les treillis de Galois Alpha utilise une approche top-down classique dans laquelle on spécialise l'intention c d'un noeud en ajoutant d'abord un nouvel attribut a puis en appliquant l'opérateur de fermeture $int \circ ext_{\alpha}$ à

$c \bigcup \{a\}$. Chaque noeud ainsi engendré doit être comparé aux noeuds déjà construits (et rangés dans une file) de manière à éviter les doublons. Nous avons expérimenté ALPHA sur un ensemble de 2274 produits informatiques, partitionnés en 59 types, extraits du catalogue C/Net. Chaque produit peut être décrit à l'aide de 234 attributs.

Dans notre première expérience nous avons construit le treillis G_{100} à partir de l'ensemble des 2274 produits (ainsi pratiquement réduit à 59 instances prototypiques), puis nous avons progressivement baissé la valeur de α et calculé les treillis G_α correspondants. Comme on peut le constater ci-dessous le nombre de noeuds (et donc le temps CPU) croît exponentiellement de 211 concepts à 165369 lorsque α varie de 100% à 91%. Il est donc ici clairement impossible d'avoir une vue complète des données au niveau des instances (c.à.d. pour α=0) et même le relâchement de la partition en classes de base (commençant avec α=100) doit être limité :

Alpha	100	98	96	94	92	91
Noeuds	211	664	8198	44021	107734	165369

Notre seconde expérience concerne la partie de G_{100} comprise entre, d'une part, le noeud dont l' extension contient les 3 classes de base (*Laptop* (252 instances pour 39 attributs impliqués dans les descriptions), *Hard-drive*(45 instances, pour 22 attributs impliqués), *Network-storage*(4 instances, pour 16 attributs impliqués)) et, d'autre part, le noeud *Bottom*. Le nouveau G_{100} contient maintenant 5 noeuds (nombre à comparer au nombre maximum de noeuds $2^3 = 8$). Le treillis G_α est calculé pour toutes les valeurs entières de α entre 100% et 0% et sa taille comparée à celle du treillis fréquent correspondant (cf. Figure 5). Nous nous intéressons d'abord aux valeurs élevées de

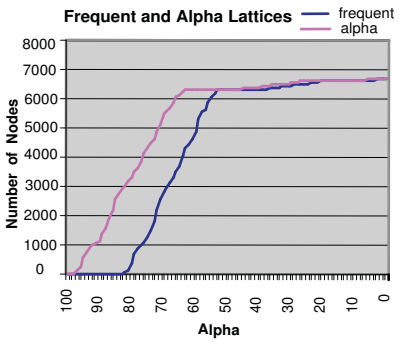

FIG. 5 – Nombre de noeuds vs Alpha pour les treillis de Galois Alpha et les treillis fréquents

α. Partant du treillis G_{100}, de nouveaux noeuds apparaissent lorsque α décroît. Ainsi pour α = 99%, un nouveau noeud apparaît sous le noeud de G_{100} représentant la classe *Laptop*. L'intention de ce nouveau noeud contient maintenant l'attribut "network-card". Ceci est dû au fait que la plupart des instances de la classe *Laptop* possède une carte réseau. Ainsi en affaiblissant la contrainte sur la classe de base nous évitons les quelques instances exceptionnelles de *Laptop* présentes dans le catalogue et qui masquaient cette propriété "par défaut" de *Laptop* dans le treillis G_{100}. De la même manière les instances de *hard-drives* sont vendues avec une maintenance (la propriété "support") . Ainsi pour

$\alpha = 92\%$, un nouveau noeud représentant les instances de *hard-drives* possédant la propriété "support" apparaît. Remarquons que dans ce cas l'attribut "support" est rare si l'on considère tous les individus ("support" apparaît dans 13 produits sur 301) et ne serait pas considéré dans un *treillis de concepts fréquents*, alors qu'il est fréquent dans la classe *hard-drive* (13 produits sur 15) et ainsi apparaît dans le treillis de Galois Alpha G_{90}. En résumé, en faisant diminuer α à partit de 100 % on a une vision plus fine des données en tenant compte de propriétés qui sont pertinentes (i.e. fréquentes) dans certaines classes de base.
Si on considère maintenant que l'on part du treillis de concepts original et que l'on fait croître lentement α de 0 vers des valeurs faibles (de l'ordre de 10%), certaines instances, *exceptionnelles* dans leur classe de base relativement à un certain terme t de $\mathcal{L}$, disparaîtront de la α-extension de t . Ces instances sont exceptionnelles car elles appartiennent à l'extension du terme t alors même que peu d'instances de cette même classe de base appartiennent à cette extension. En conséquence, certaines propriétés très rares dans certaines classes de base, ne seront plus utilisables pour distinguer les concepts. Par exemple, quelques *Laptops* seulement ont la propriété "Digital-Signal-Protocol", et ainsi pour $\alpha = 6\%$, les noeuds dont l'intention contient la propriété "Digital Signal Protocol" ne contiennent plus d'instances de *Laptop* dans leur extension. Ainsi les termes de la forme $\{$Laptop, Digital-Signal-Protocol, ... , $\}$ deviennent équivalents au noeud $Bottom$, alors que d'autres termes incluant " Digital-Signal-Protocol" deviennent équivalents (car leur extensions ne diffèrent que par la présence d'instances de *Laptop*). L'effet résultant est une diminution du nombre de noeuds. Une lecture plus précise de la Figure 5 montre qu'il peut y avoir beaucoup de noeuds même pour des valeurs élevées de α. Dans cet exemple particulier ceci est du au fait qu'une classe de base, *Laptop*, a un treillis fréquent qui envahit le treillis Alpha. Une expérience menée sur une partie seulement des données du catalogue (24 classes de base représentant en tout 1187 objets) obtenues en retirant certaines classes envahissantes, montre que le treillis Alpha résultant est plus détaillé (pour une même valeur α) que le treillis fréquent correspondant (voir aussi Figure 6) :

Alpha	100	80	50	30	0
Noeuds Alpha	158	842	1493	1900	2202
Noeuds Frequent	2	18	18	50	2202

5 Le point de vue Alpha

5.1 Règles d'implication Alpha

Les règles d'association, dans le domaine de la fouille de donnée, sont des implications dont la valeur de vérité est observée sur un ensemble d'instances I. Chaque règle est associée à une valeur de *support* (la fréquence de sa partie prémisse dans I), et à une valeur de *confiance*. Lorsque la confiance est fixée à 1, les règles sont dites *règles d'implication*. Lorsque l'on considère les treillis de concepts, l'ordre partiel induit sur les termes du langage par la correspondance de Galois, correspond à un ensemble de règles d'implication. Plus précisément, $T_2 \preceq_I T_1$ signifie que $ext_I(T_1) \subseteq ext_I(T_2)$ i.e.

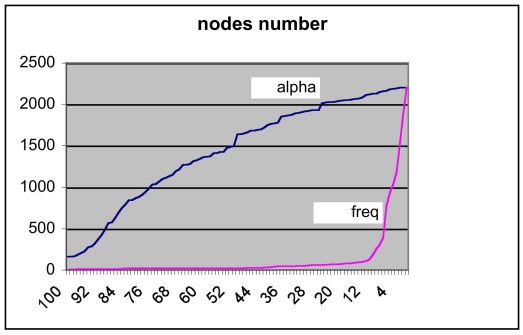

FIG. 6 – Nombre de noeuds vs Alpha pour les treillis de Galois Alpha et les treillis fréquents

l'implication logique $T_1 \rightarrow T_2$ est vraie sur I. Pour une telle règle, on appelera T_1 la partie gauche (ou prémisse) et T_2 la partie droite (ou conclusion) de la règle.

Les règles d'association correspondant à un ensemble d'instances I sont souvent construites en deux étapes : on construit le treillis de concepts associé à I, puis on extrait du treillis les règles d'association. L'idée générale est qu'un noeud d'un treillis de concepts correspond à une classe d'équivalence de termes partageant tous la même extension. L'intention du noeud, c.à.d. son plus grand élément (le plus spécifique), a donc la même extension que les termes les plus petits (les plus généraux) de la classe d'équivalence. En prenant comme partie gauche un terme minimal et comme partie droite un terme fermé on obtient un ensemble de règles d'implications. Une partie de cet ensemble forme la base de Guigues-Duquenne des règles d'implications associées à I (Guigues & Duquenne, 1986). Pour rendre les règles plus lisibles, la partie droite de la règle est retirée de la partie gauche. Par exemple, considérons le noeud $(abc, \{i1, i2, i3\})$ et supposons que les plus petits termes de la classe d' équivalence sont $\{a, b\}$, avec $ext_I(a)$= $ext_I(b)$ = $\{i1, i2, i3\}$. On obtient alors les règles d'implication $\{a \rightarrow abc, b \rightarrow abc\}$ qui se réécrivent $\{a \rightarrow bc, b \rightarrow ac\}$. Dans les treillis fréquents, l'extension d'un terme est redéfinie comme vide lorsque le terme n'est pas fréquent dans I, c'est à dire quand son extension contient moins de $minsupport * |I|$ instances de I. En consequence les règles d' implication extraites des noeuds restants ont toutes un support plus grand que $minsupport$.

En ce qui concerne les treillis de Galois Alpha, $T_2 \preceq_\alpha T_1$ signifie que $ext_\alpha(T_1) \subseteq ext_\alpha(T_2)$ et nous dirons que la $\alpha - implication$ $T_1 \rightarrow_\alpha T_2$ est vraie sur le couple $(I, \mathcal{BC})$. Ces règles dérivant d'un treillis de Galois, , c.à.d. correspondant à l'inclusion de l'extension de la partie gauche dans l'extension de la partie droite, les $\alpha - implications$ ont les propriétés suivantes :

-Si $T_1 \rightarrow_\alpha T_2$ et $T_2 \rightarrow_\alpha T_3$, alors $T_1 \rightarrow_\alpha T_3$ (Transitivité)

-SI $T_1 \rightarrow_\alpha T_2$, et $T_1 \preceq$ T, alors $T \rightarrow_\alpha T_2$ (Monotonie)

-SI $T_1 \rightarrow_\alpha T_2$ et $T_1' \rightarrow_\alpha T_2'$, alors $T_1 \cup T_1' \rightarrow_\alpha T_2 \cup T_2'$ (Additivité)

De plus on dispose du *modus ponens* en tant que règle d'inférence :

Si $i\ isa_\alpha\ T_1$ et $T_1 \rightarrow_\alpha T_2$, alors $i\ isa_\alpha\ T_2$

La propriété d'additivité au niveau des instances s'écrit alors comme suit :

Si $i\ isa_\alpha\ T_1$ et $i\ isa_\alpha T_2$ et $BCl(i) sat_\alpha T_1 \cup T_2$ alors $i\ isa_\alpha T_1 \cup T_2$

Nous n'approfondirons pas ici les $\alpha - implications$ et la construction d'une base de telles règles d'une manière semblable à la construction de la base de Guigues-Duquenne. Nous voudrions cependant illustrer sur un exemple ce en quoi elles permettent d'exprimer des notions d'exceptions lorsque les individus sont étiquetés par des types.

Pour cela supposons que nous séparions des espèces animales (chacune correspondant à une instance) en les classes de base suivantes : *mammifères, oiseaux, insectes*. Nous cherchons à extraire des règles générales de ces données. Une règle intuitive est par exemple : "un animal qui *vole* devrait avoir des *ailes*. Cette implication est vraie aussi bien pour les oiseaux (les oiseaux coureurs, comme l'autruche, ne contredisent pas la règle) que pour les insectes. Pour être universelle, la règle devrait aussi être vraie pour les mammifères, qui en général ne volent pas, mais est contredite par l'écureuil volant. L'approche Alpha ici permet de tirer parti de ce que peu de mammifères volent (en d'autre termes, la prémisse de la règle n'est pas fréquente dans la classe *mammifère* à laquelle appartient l'individu qui falsifie la règle). Utiliser une α-extension permet de retirer l'écureuil-volant de l'extension de la prémisse de la règle. Ici, une valeur faible de α est suffisante pour obtenir une règle d' $\alpha - implication$ (avec donc une confiance de 1) exprimant que les animaux volant ont des ailes. Bien sûr des valeurs plus élevées de α, (proches de 100) évitent aussi la falsification de cette règle. Cependant dans ce dernier cas, une règle d' α-implication exprime quelque chose de différent : elle ne s'applique à un individu que lorsque la prémisse de la règle est commune à la *plupart* des individus de la même classe de base. Dans notre exemple, seuls les *oiseaux* seraient ainsi concernés par une telle règle mais pas les *insectes*.

5.2 Alpha extensions et "Rough sets"

Nous traitons ici du point de vue ensembliste de ce travail. Nous considérons qu'un individu i α-appartient à un sous-ensemble e de I (nous noterons $i\ \in_\alpha\ e$) lorsque i appartient à $proj_\alpha(e)$. Dans ce cas $i\ isa_\alpha\ T$ s'écrit $i \in_\alpha ext(T)$. Un autre point de vue ensembliste traite de données partitionnées *a priori* en classes de base : il s'agit de la théorie des ensembles *rugueux* (la théorie des *rough sets* ((Pawlak, 1996)). Dans cette théorie, tout sous-ensemble e de I a un plus grand minorant $inf(e)$ et un plus petit majorant $sup(e)$ pris parmi les réunions de classes de base. Si on compare la vue ensembliste Alpha avec celle des ensembles rugueux, on constate immédiatement que pour tout sous-ensemble e de I, on a $proj_{100}(e) = inf(e)$. Dans la théorie des ensembles rugueux l'appartenance rugueuse $\mu_e(i)$ est la proportion d'individus de la même classe de base que i qui appartiennent à e. Pour comparer ces deux visions ensemblistes nous utilisons α pour discrétiser l'appartenance μ. Ceci conduit à la relation d'appartenance suivante :

Définition 12
Soient i un élément de I et e un sous-ensemble de I. La fonction d'appartenance rugueuse seuillée $\in^R_\alpha$ se définit comme suit :
$i \in^R_\alpha e$ *ssi* $\mu_e(i) \geq \frac{\alpha}{100}$

Remarquons que lorsque $i \notin ext(T)$, on a $i \notin_\alpha ext(T)$, alors qu'il est parfaitement possible d'avoir $i \in_\alpha^R ext(T)$. En effet le partitionnement sur les ensembles rugueux exprime une indiscernabilité entre individus de la même classe de base. Ainsi dans la théorie des ensembles rugueux il y a un certain degré d'appartenance de i à e dès que $e \bigcap Bc(i)$ n'est pas vide. Au contraire, du point de vue ensembliste Alpha, l'appartenance de i à un ensemble e est un pré-requis pour la α-appartenance. La α-appartenance exprime donc à quel point l'appartenance à e est fréquente *aussi* parmi les individus de la même classe de base que i : si i est exceptionnel relativement à un terme t, alors i n'appartient pas à $ext_\alpha(t)$ même si i appartient à l'extension classique de t. Une consequence heureuse de la vue ensembliste Alpha est l'opportunité de construire des treillis de Galois. Il est aisé de voir que pour $\alpha \neq 100$ la fonction ext correspondant à la vue ensembliste rugueuse (donc utilisant $\in_\alpha^R$) ne forme pas, avec la fonction int, une correspondance de Galois (voir la condition C3 dans la définition 2).

6 Travaux proches, conclusion et perspectives

Des travaux récents en Représentation des Connaissances et en Apprentissage par Machine se sont intéressés aux correspondances et structures de Galois avec des langages de termes plus complexes que les ensembles d'attribut. (Ganter & Wille, 1999; Ganascia, 1993; Liquiere & Sallantin, 1998; Ganter & Kuznetsov, 2001). Nous avons vu ci-dessus qu'en modifiant la notion d'extension en se référant à une partition de l'ensemble des instances I, on modifie le treillis des extensions qui n'est plus $\mathcal{P}(I)$ et on obtient une nouvelle famille de treillis de Galois. En particulier nous avons vu que les treillis Iceberg (ou treillis de concepts fréquents) (Waiyamai & Lakhal, 2000; Stumme *et al.*, 2002) sont formellement des treillis de Galois Alpha particuliers pour lesquels tous les individus appartiennent à la même et unique classe de base. D'autre part, les règles d' implication associées aux treillis Alpha correspondent à l'inclusion des α-extensions et une base canonique de telles $\alpha - implications$ pourrait être extraite d'un treillis Alpha de la même manière qu'une base canonique de règles d'implications peut être extraite d'un treillis de concepts fréquents. Remarquons que les $\alpha - implications$ héritent de la structure de treillis de Galois des propriétés intéressantes (comme la transitivité) inhabituelles lorsqu'on travaille avec des règles "approximatives".

En ce qui concerne la construction des treillis Alpha, il serait intéressant d'adapter des algorithmes efficaces de construction de treillis de concepts (e.g. (Kuznetsov & Obiedkov, 2002)). Cependant, la propriété 3 conduit à une autre manière d'engendrer les treillis de Galois Alpha en construisant d'abord les treillis fréquents correspondant à chaque classe de base, puis en les fusionnant à l'aide d'un opérateur de *subposition*. Un tel opérateur a été récemment proposé par (Valtchev *et al.*, 2002) pour construire et maintenir efficacement un treillis de concepts. Remarquons que dans notre cas ce procédé de construction est la base d'une *incrémentalité par classes de base* des treillis de Galois Alpha. Pour conclure il y a encore beaucoup à faire pour expérimenter et examiner les problèmes théoriques et pratiques concernant les treillis Alpha et les α-implications correspondantes. Cependant nous pensons que cette structure représente un outil flexible d'investigation des données permettant de traiter des exceptions relatives à un point de vue préliminaire des données associé à un typage. Ce point de vue

préliminaire est une partie du biais que tout utilisateur averti applique pour extraire et apprendre des connaissances à partir de données.

Remerciements. Nous remercions Nathalie Pernelle pour sa contribution, au travers de nombreuses discussions, au travail présenté ici, et aussi Philippe Dague pour avoir patiemment relu ce manuscrit.

Références

BIRKHOFF G. (1973). *Lattice Theory*. Rhode Island : American Mathematical Society Colloquium Publications.

BOCK H. & DIDAY E. (2000). *Analysis of Symbolic Data*. H.H. Bock and E. Diday, Springer Verlag.

GANASCIA J. (1993). Tdis : an algebraic formalization. In *Int. Joint Conf. on Art. Int.*, volume 2, p. 1008–1013.

GANTER B. & KUZNETSOV S. O. (2001). Pattern structures and their projections. *Proceeding of ICCS-01, LNCS*, **2120**, 129–142.

GANTER B. & WILLE R. (1999). *Formal Concept Analysis : Logical Foundations*. Springer Verlag.

GUIGUES J. & DUQUENNE V. (1986). Famille non redondante d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences humaines*, **95**, 5–18.

HERETH J., STUMME G., WILLE R. & WILLE U. (2000). Conceptual knowledge discovery and data analysis. In *International Conference on Conceptual Structures*, p. 421–437.

KUZNETSOV S. & OBIEDKOV S. (2002). Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theoritical Artificial Intelligence*, **2/3**(14), 189–216.

LIQUIERE M. & SALLANTIN J. (1998). Structural machine learning with galois lattice and graphs. In *ICML98, Morgan Kaufmann*.

PAWLAK Z. (1996). Rough sets, rough relations and rough functions. *Fundamenta Informaticae*, **27**(2/3), 103–108.

PERNELLE N., ROUSSET M.-C., SOLDANO H. & VENTOS V. (2002). Zoom : a nested galois lattices-based system for conceptual clustering. *J. of Experimental and Theoritical Artificial Intelligence*, **2/3**(14), 157–187.

PERNELLE N., ROUSSET M.-C. & VENTOS V. (2001). Automatic construction and refinement of a class hierarchy over multi-valued data. In *5th Conference on Principles and practice of Knowlewdge Discovery in Databases, PKDD'01, Lecture Notes in Artificial Intelligence*, p. 386–398 : Springer-Verlag.

STUMME G., TAOUIL R., BASTIDE Y., PASQUIER N. & LAKHAL L. (2002). Computing iceberg concept lattices with titanic. *Data and Knowledge Engineering*, **42**(2), 189–222.

VALTCHEV P., MISSAOUI R. & LEBRUN P. (2002). A partition-based approach towards building galois (concept) lattices. *Discrete Mathematics*, **256**(3), 801–829.

WAIYAMAI K. & LAKHAL L. (2000). Knowledge discovery from very large databases using frequent concept lattices. In *11th European Conference on Machine Learning, ECML'2000*, p. 437–445.