Méthodes de classification conceptuelle basées sur les treillis de Galois et applications

(Paru dans Revue d'Intelligence Artificielle, 1995, 9(2),pp.105-137)

Robert Godin[1], Guy Mineau[2], Rokia Missaoui[1], Hafedh Mili[1]

1 - Département d'Informatique
Université du Québec à Montréal
C.P.8888, Succursale Centre Ville
Montréal, Québec
Canada, H3C 3P8
Adresse électronique: godin.robert@uqam.ca

2- Département d'Informatique
Faculté des Sciences et de Génie
Université Laval
Québec, Québec
Canada, G1K 7P4
Adresse électronique: mineau@ift.ulaval.ca

RÉSUMÉ. Diverses structures utilisées dans des méthodes de classification conceptuelle sont présentées dans un cadre unifié à partir de la notion de treillis de Galois (treillis de concepts). La présentation dans un cadre unificateur permet de faire ressortir les points communs entre les méthodes et de considérer une variété de combinaisons potentielles. L'utilisation de ces structures est illustrée par diverses d'applications: recherche documentaire, réutilisation, conception des hiérarchies de classes, génération de règles d'implication à partir de bases de données, acquisition et organisation des connaissances.

ABSTRACT. Several structures used for conceptual clustering are presented in a uniform framework based on the notion of Galois lattices (concept lattices). The unified framework reveals the common aspects of the methods and leads to consider a variety of potential combinations. The usefulness of these structures is illustrated by several applications: information retrieval, software reuse, design of class hierarchies, data mining, knowledge acquisition and organization.

MOTS-CLÉS: classification conceptuelle, formation de concepts, treillis de Galois, treillis de concepts, applications.
KEY WORDS: conceptual clustering, concept formation, Galois lattice, concept lattice, applications.

1. Introduction

La notion de treillis de Galois d'une relation (ou treillis de concepts) est à la base d'une famille de méthodes de classification conceptuelle. Introduite par Barbut et Monjardet [Barbut 1970], cette approche a été popularisée par Wille qui a utilisé la notion de treillis de Galois comme base de l'analyse formelle de concepts [Wille 1982]. Cette approche a initialement trouvé des applications en intelligence artificielle pour la représentation et l'acquisition de connaissances [Ganter 1986, Wille 1989]. Wille propose de considérer chaque élément du treillis comme un concept formel et le graphe (diagramme de Hasse) comme une relation de généralisation/spécialisation entre les concepts. Le treillis est donc perçu comme une hiérarchie de concepts. Chaque concept est une paire composée d'une extension représentant un sous-ensemble des instances de l'application et d'une intention représentant les propriétés communes aux instances.

Du point de vue du domaine de l'apprentissage machine, la construction automatique d'une telle hiérarchie peut être vue comme une méthode de *classification (regroupement) conceptuelle* [Michalski 1981]. Comme des algorithmes incrémentaux efficaces ont été développés, sa construction incrémentale peut être considérée comme une *méthode incrémentale de formation de concepts* [Fisher 1991]. Ces catégories de méthodes font partie de l'apprentissage *non supervisé* parce que les concepts ne sont pas prédéterminés et les instances utilisées pour l'apprentissage ne sont pas pré-classifiées [Carbonell 1990]. Contrairement aux hiérarchies produites par les autres méthodes de même catégorie, tel que UNIMEM [Lebowitz 1987], COBWEB [Fisher 1987] et CLASSIT [Gennari 1990] , le treillis de Galois a une structure formellement définie qui ne dépend pas de divers paramètres, de l'ordonnancement des instances ou de particularités algorithmiques. Une autre caractéristique distinctive par rapport à ces méthodes est que la hiérarchie n'est pas nécessairement réduite à une structure d'arbre. La hiérarchie obtenue met en évidence de façon exhaustive les regroupements potentiellement intéressants par rapport aux observations. En contraste avec les méthodes habituelles de formation de concepts qui cherchent à produire un nombre limité de concepts en optimisant le contraste et la cohérence entre les classes, le nombre de concepts dans un treillis de Galois est souvent plus grand et la qualité des concepts plus variable. Dans des domaines d'application où les propriétés des instances sont très nombreuses et peu fiables, ceci peut conduire à la formation de nombreux concepts peu pertinents.

De nombreux développements ont été réalisés au cours des dernières années concernant entre autres:

- la visualisation du treillis [Godin 1989, Wille 1984],
- la simplification et l'élagage du treillis par décompositions ou par méthodes heuristiques [Anquetil 1994, Ganter 1989, Godin 1993, Liquière 1990].
- son utilisation dans un processus interactif d'acquisition de connaissances [Wille 1989],
- la génération de règles à partir du treillis [Godin 1994, Guigues 1986, Wille 1992],

- l'extension à des représentations plus riches telles l'utilisation d'attributs [Ganter 1986, Godin 1986], de graphes conceptuels [Mineau 1990, Mineau 1994] et de connaissances taxinomiques [Godin 1993, Mineau 1994],

- les algorithmes de construction du treillis [Bordat 1986, Ganter 1984, Guénoche 1990], les algorithmes incrémentaux de construction du treillis et des règles [Carpineto 1993, Godin 1994, Godin 1995] et les algorithmes de construction de diverses variantes [Dicky 1994, Godin 1993, Mineau 1994].

Carpineto et Romano [Carpineto 1993] ont démontré l'utilité du treillis pour la découverte et la prédiction de classes à partir de plusieurs corpus de données du domaine de l'apprentissage machine. Parmi les publications récentes dans le domaine, notons aussi [Daniel-Vatonne 1993, Guénoche 1993].

Le treillis de Galois a aussi été proposé comme médium pour la recherche documentaire [Godin 1993, Godin 1993]. Dans ce contexte, le treillis est généré à partir de la relation d'indexage entre documents et termes d'index. Le diagramme de Hasse est utilisé comme structure de base pour supporter une interface de navigation permettant à l'utilisateur de graduellement élargir ou spécialiser sa requête en terme des sous-ensembles de documents et de termes présents dans le treillis. Dans une optique similaire, la notion d'espace de connaissance a été proposée comme support au repérage d'images représentées par des graphes conceptuels [Mineau 1992].

Récemment, des applications au domaine d'ingénierie du logiciel ont été considérées. Krone et Snelting [Krone 1994] ont utilisé l'analyse de concepts pour étudier la structure des configurations à partir d'informations extraites du code source. Dans un vaste projet de développement d'outils pour un atelier de génie du logiciel [Godin 1995], nous avons utilisé le treillis de Galois et une variante, le treillis élagué, pour la réutilisation. Le regroupement conceptuel sert comme outil de repérage et d'abstraction d'artefacts plus génériques avec une meilleure réutilisabilité. Dans la même veine, une approche plus ambitieuse à la réutilisation consiste à identifier des familles larges de composantes reliées entre elles qui peuvent être extraites et réutilisées en bloc. Dans [Mineau 1993], la notion d'espace de connaissance est appliquée pour induire une hiérarchie de modèles génériques à partir de modèles existants. Le regroupement conceptuel devient un outil d'assistance à l'analyse du domaine, discipline qui est de plus en plus considérée comme primordiale au processus de réutilisation [Prieto-Diaz 1991b]. Dans le contexte du développement orienté objet, le treillis de Galois et le treillis élagué peuvent servir comme base pour des outils de génération et de réorganisation des hiérarchies de classes à partir de la spécification de leurs propriétés [Dicky 1994, Godin 1993].

Dans la section suivante de cet article, nous présentons diverses structures à la base des méthodes de classification conceptuelle utilisées dans ces applications. La présentation est faite dans un cadre unificateur permettant de faire ressortir les points communs entre les structures. Par la suite, dans la section 3, nous faisons sommairement un état de travaux sur les applications de ces méthodes en faisant ressortir les structures utilisées par rapport au cadre établi dans la section 2.

2. Treillis de Galois et structures dérivées

Cette section rappelle les définitions de base de treillis de Galois et présente dans un cadre unifié un certains nombre de structures alternatives dérivées qui ont été utilisées pour la classification.

2.1. *Treillis de Galois*

Cette sous-section rappelle la définition de base de treillis de Galois d'une relation binaire et quelques propriétés essentielles. Pour une couverture plus détaillée des aspects formels voir [Barbut 1970, Davey 1992, Wille 1992]

Soit deux ensembles finis, E et E', et une relation binaire, $R \subseteq E \times E'$ (Tableau 1), entre ces deux ensembles, on peut représenter par un treillis les regroupements naturels des éléments de E et de E' par rapport à la relation R (Figure 1). Cette structure est appelée treillis de Galois [Barbut 1970] ou treillis de concepts [Wille 1992]. Chaque élément du treillis est un couple (appelé concept formel par Wille) noté, (X,X'), et composé d'un ensemble $X \in P(E)$ et d'un ensemble $X' \in P(E')$[1] satisfaisant aux deux propriétés suivantes:

1) $X' = f(X)$ où $f(X) = \{x' \in E' \mid \forall x \in X, xRx'\}$
2) $X = f'(X')$ où $f'(X') = \{x \in E \mid \forall x' \in X', xRx'\}$.

Quoique la définition soit totalement symétrique, dans les applications, on associe habituellement les éléments de E à des objets de l'application et ceux de E' aux propriétés de ces objets. En apprentissage machine, on parlera d'instances ou d'exemples et de leurs propriétés ou attributs. A noter que par définition, $f(\emptyset) = E'$ and $f'(\emptyset) = E$. Les couples satisfaisants les deux propriétés précédentes sont dits *complets*. Le *treillis de Galois* (G) d'une relation binaire R est l'ensemble de tous les couples complets dérivés de R. Le terme *complet* représente le fait que pour un couple (X,X'), s'il y a des éléments de E qui ne sont pas dans X mais qui sont reliés à tous les éléments de X', alors ils doivent être ajoutés à X pour que le couple soit complet. À cause de la symétrie entre X et X' dans la définition, le même raisonnement s'applique à X'. Cette idée de complétude des couples est formalisée par la notion mathématique de *fermeture* dans les ensembles ordonnés. Une fermeture dans un ensemble ordonné $(E,\geq)$ est une application, $h: E \rightarrow E$, ayant les propriétés suivantes:

i) $\forall x \forall y,\ x \geq y \Rightarrow h(x) \geq h(y)$
ii) $\forall x,\ h(x) \geq x$
iii) $\forall x,\ h(h(x)) = h(x)$.

L'image $h(x)$ de x dans E est la *h-fermeture* de x; si $x = h(x)$, x est *h-fermé* ou un élément fermé pour h. Les applications $h = f \circ f'$ et $h' = f' \circ f$ sont respectivement des fermetures dans E et E'. L'ensemble des éléments h-fermés de E correspond aux ensembles X des couples complets de G et forme un treillis H isomorphe à G. Symétriquement, les éléments h'-fermé de E' sont les ensembles X' des couples complets et forment aussi un treillis H' isomorphe à G.

---

[1] P(S) représente l'ensemble des parties de S.

Une relation d'ordre naturelle entre les couples du treillis est définie par rapport à l'inclusion des ensembles X ou X'. Soit $C_1=(X_1,X'_1)$ et $C_2=(X_2,X'_2)$,

$$C_1 < C_2 <=> X'_1 \subset X'_2 .$$

Tableau 1. *Représentation matricielle de la relation R.*

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

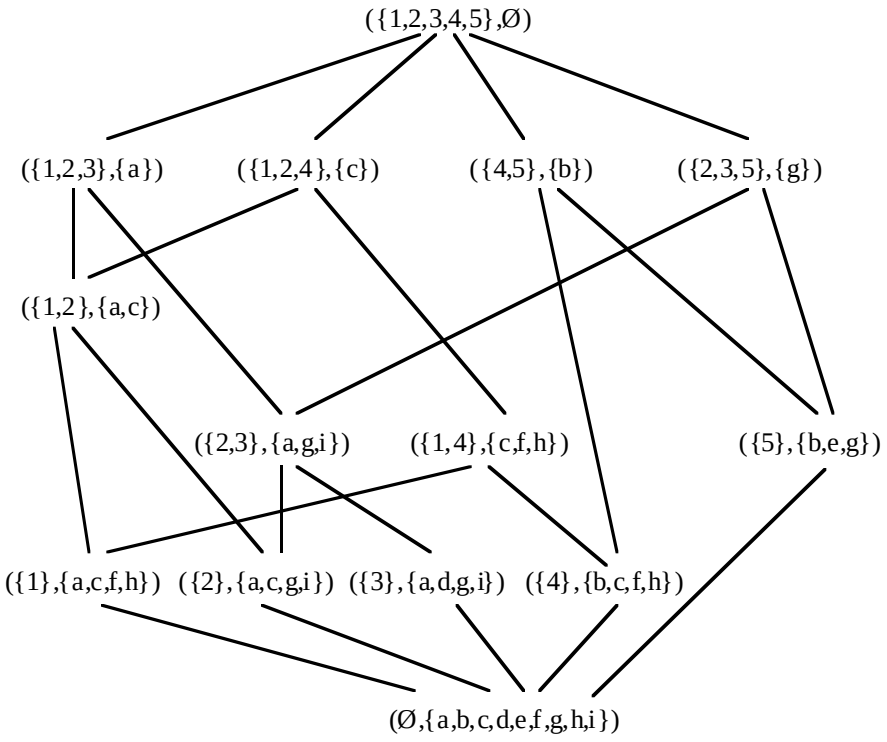

Figure 1. *Treillis de Galois correspondant à la relation du Tableau 1.*

Il y a une relation duale entre les ensembles X et X' étant donné que,

$$X'_1 \subset X'_2 <=> X_2 \subset X_1$$

et en conséquence,

$$C_1 < C_2 <=> X_2 \subset X_1.$$

La relation d'ordre partiel entre les couples est utilisée pour générer le graphe du treillis de la façon suivante: il y a un arc (C_1,C_2) si $C_1 < C_2$ et il n'existe aucun autre couple C_3 dans G tel que $C_1 < C_3 < C_2$. C_1 est appelé *parent* de C_2 et C_2 *enfant* de C_1. Lorsque l'on dessine le graphe (diagramme de Hasse) la direction des arcs est par convention toujours dans le même sens, soit vers le haut ou vers le bas. Le graphe peut être interprété comme une représentation de la relation de généralisation/spécialisation entre les couples où $C_1 < C_2$ représente le fait que C_1 est plus général que C_2. Soit C, un sous-ensemble d'éléments de G, inf(C) et sup(C) dénotent respectivement la borne inférieure (plus grand minorant de C) et borne supérieure (plus petit majorant) des éléments dans C. L'appellation treillis de Galois vient d'une part du fait que l'ensemble des couples complets forme toujours un treillis, ce qui garantie l'existence de inf(C) et de sup(C). D'autre part le couple (f,f') représente une *correspondance de Galois* entre P(E) et P(E') et le treillis est une représentation naturelle de cette correspondance.

Dans le pire cas, le treillis de Galois peut croître exponentiellement avec le nombre d'objets et de propriétés. Cependant, lorsqu'il y a une borne supérieure, disons k, sur $|f(\{x\})|$, ce qui est le cas dans les applications habituellement considérées, la taille du treillis est bornée linéairement par rapport à $|E| = n$ [Godin 1986] :

$$|G| \leq 2^k n.$$

De plus, malgré le facteur exponentiel en k, des expériences avec de nombreuses applications et des résultats théoriques dans un cadre de distribution uniforme montrent que le rapport |G|/n est assez stable et de beaucoup inférieur à 2^k. En pratique, dans des applications de recherche documentaire, nous avons toujours observé que

$$|G| \leq k' n,$$

où k' est la valeur moyenne de $|f(\{x\})|$. De plus, des estimations théoriques toujours dans un cadre de distribution uniforme suggèrent une croissance linéaire par rapport à k'. Pour plus de détails sur la complexité du treillis voir [Godin 1989, Godin 1993].

2.2. *Treillis d'héritage*

La structure de treillis de Galois comporte beaucoup de redondance. Pour plusieurs applications, une structure alternative que nous appelons *treillis d'héritage* permet de représenter la même information de façon non redondante et beaucoup plus compacte. Cette représentation est souvent plus utile parce qu'elle fait ressortir les différences entre les couples du treillis.

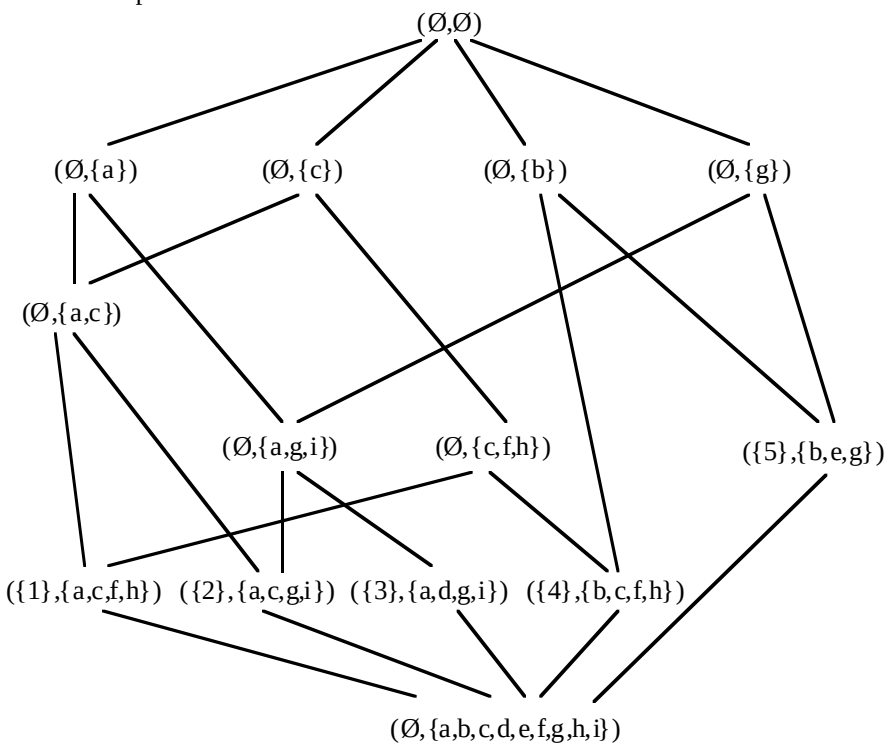

Figure 2. *Treillis d'héritage selon X (G-HX).*

Pour un couple complet quelconque $C = (X, X')$, X apparaît dans tous les ancêtres de C et symétriquement, X' apparaît dans tous ses descendants. On peut donc éliminer ces éléments redondants sans perdre d'information si la structure du graphe est maintenue. Les éléments de X ainsi éliminés sont alors retrouvés par héritage par rapport aux ancêtres. Les éléments de X' sont retrouvés par héritage par rapport aux descendants. Soit :

$r(X) = \{x \in E \mid x \in f'(X')$ et il n'existe pas de $C' = (Y,Y') > C$ tel que $x \in Y\}$
$r(X') = \{x' \in E' \mid x' \in f(X)$ et il n'existe pas de $C' = (Y,Y') < C$ tel que $x' \in Y'\}$

En d'autres termes, $r(X)$ représente les éléments non redondants de X, et $r(X')$ les éléments non redondants de X'. Les définitions suivantes sont équivalentes et font

ressortir le fait que $x \in r(X)$ si C est le couple le plus général de G qui contient x, et $x' \in r(X')$ si C est le couple le plus spécifique qui contient x' :

$$r(X) = \{x \in E \mid x \in f'(X') \text{ et } C = \sup \{(Y,Y') \in G \mid x \in Y\}\}$$
$$r(X') = \{x' \in E' \mid x' \in f(X) \text{ et } C = \inf \{(Y,Y') \in G \mid x' \in Y'\}\}.$$

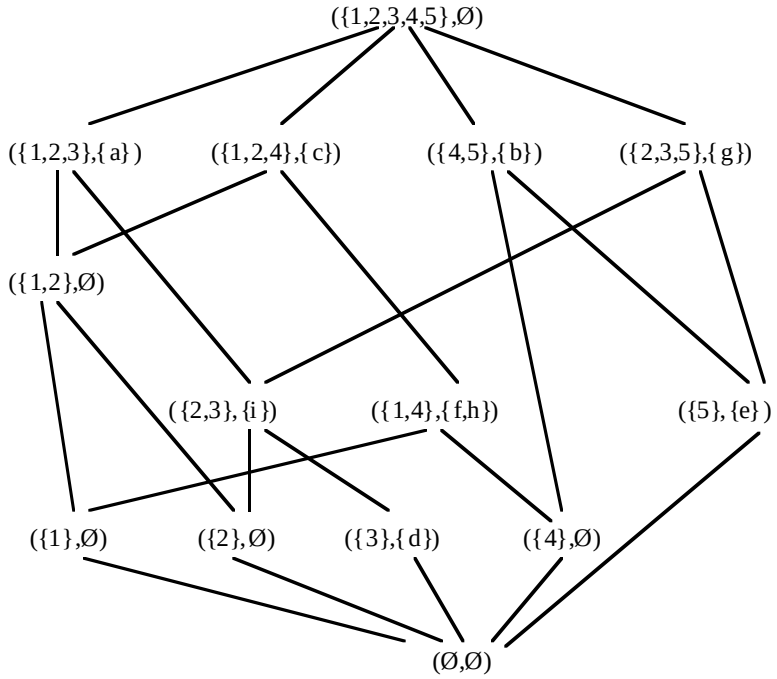

Figure 3. *Treillis d'héritage selon X' (G-HX').*

On peut aussi déduire que les éléments de $r(X)$ sont ceux qui sont exactement en relation avec X' mais avec aucun autre élément; et que symétriquement, les éléments de $r(X')$ sont en relation avec X mais avec aucun autre élément :

$$r(X) = \{x \in E \mid f(\{x\}) = X'\}$$
$$r(X') = \{x' \in E' \mid f'(\{x'\}) = X\}.$$

On peut vouloir appliquer l'héritage sélectivement pour les ensembles X ou X'. On définit donc un treillis d'héritage selon X (*G-HX*) comme l'ensemble des couples $(r(X), X')$ (Figure 2), et un treillis d'héritage selon X' (*G-HX'*) comme l'ensemble des couples $(X, r(X'))$ (Figure 3) et un treillis d'héritage (*G-H*) comme l'ensemble des couples $(r(X), r(X'))$ (Figure 4). Pour un couple C, les valeurs de X (respectivement X') peuvent être calculées en prenant l'union des ensembles $r(X)$ (respectivement $r(X')$) pour les descendants (respectivement ancêtres) de C, incluant C lui-même.

On obtient donc trois représentations alternatives du treillis. Suivant l'application, les contraintes de performance en temps par rapport à l'espace mémoire, ces structures peuvent être considérées comme représentations alternatives par rapport à la représentation complète. Dans ses travaux, Wille [Wille 1992] utilise d'ailleurs toujours le treillis d'héritage dans les représentations graphiques des treillis sans toutefois faire la distinction de façon explicite.

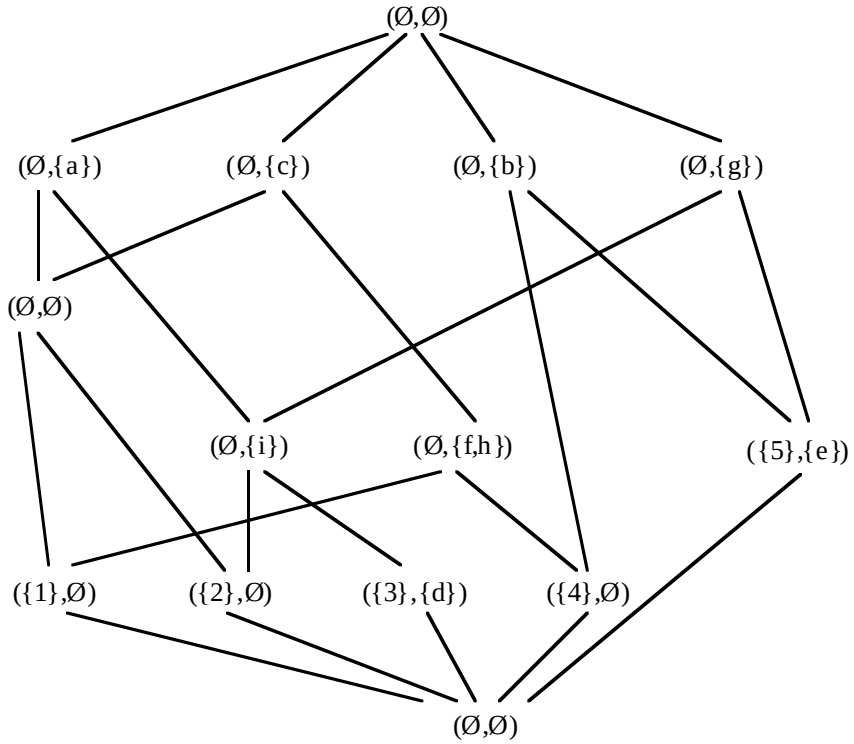

Figure 4. *Treillis d'héritage (G-H).*

2.3. *Treillis de Galois élagué*

Une sous-structure du treillis de Galois, que nous appelons treillis de Galois élagué, a été utilisée dans plusieurs applications. Un avantage de cette structure est la complexité de beaucoup inférieure pour certaines configurations de données. Cette structure a été proposée initialement dans un contexte plus général de classification de graphes conceptuels [Mineau 1990]. Cook [Cook 1992] a utilisé cette structure sans la nommer ainsi pour représenter la hiérarchie d'interface d'une librairie objet. La présentation suivante exprime le concept dans le contexte d'une relation binaire.

Le treillis de Galois élagué est l'ensemble des couples complets pour lesquels $(r(X), r(X')) \neq (Ø,Ø)$. Dans certaines applications, on peut vouloir éliminer même les couples pour lesquels soit $r(X)$, soit $r(X')$ est vide. On parlera de treillis élagué selon

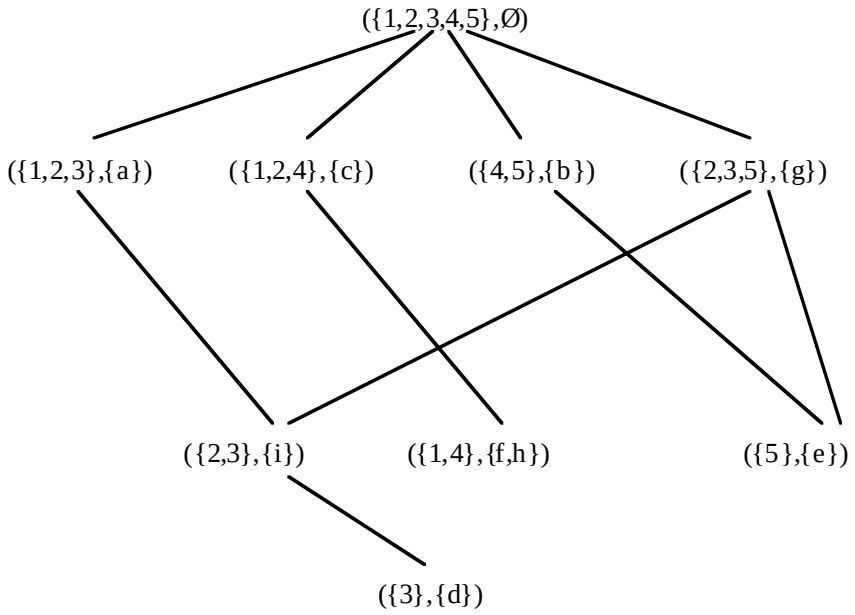

Figure 5. *Treillis d'héritage selon X' élagué selon X' (G-HX'/X') pour la relation du Tableau 1.*

X ou de treillis élagué selon *X'*. À noter que la structure résultante n'est pas nécessairement un treillis. On peut évidemment combiner l'héritage avec ces notions pour obtenir diverses combinaisons. En tout, il y a quatre alternatives pour le choix des couples (treillis de Galois sans élagage, treillis élagué, élagué selon *X*, élagué selon *X'*) et quatre représentations possibles (sans héritage, avec héritage, avec héritage selon *X*, avec héritage selon *X'*), pour un total de 16 combinaisons possibles. La Figure 5 montre une combinaison utilisée dans certaines applications, soit le treillis d'héritage selon *X'* élagué selon *X'* (*G-HX'/X'*). En combinant les définitions, on obtient que *G-HX'/X'* est l'ensemble des couples (*X*,*r*(*X'*)) satisfaisant:

1) $X = f'(r(X'))$
2) $r(X') = \{x' \in E' \mid f'(\{x'\}) = X\}$.

On peut démontrer que le nombre de couples dans *G-HX'/X'* est borné par $|E'|$ [Mineau 1994]. Avec une borne supérieure *k* sur $|f(\{x\})|$, $|E'|$ est bornée par *kn* et en conséquence :

$|G\text{-}HX'/X'| \leq k$ n.

Dans plusieurs applications, la croissance asymptotique de $|G\text{-}HX'/X'|$ devient logarithmique en n conformément à la loi de Bradford-Zipf [Brookes 1977]. Cette loi est applicable dans plusieurs phénomènes tel que la croissance d'un vocabulaire par rapport à la longueur d'un texte ou la croissance d'un vocabulaire d'indexation par rapport au nombre total de documents. Dans plusieurs applications, nous avons observé que la taille du treillis élagué est de beaucoup inférieure au treillis de Galois.

Dans le treillis élagué, le nombre d'ancêtres d'un couple est borné par k parce que chaque élément dans $f(\{x\})$ apparaît dans exactement un noeud et chaque ancêtre d'un couple contenant x contient au moins un élément de $f(\{x\})$. Donc, le nombre d'ancêtres est borné par k.

D'autre part, le treillis élagué peut être généré beaucoup plus rapidement que le treillis de Galois pour certaines configurations de données [Godin 1995].

2.4. *Extension à une représentation avec attributs*

Les éléments de E' dans les définitions précédentes sont des valeurs atomiques. Dans plusieurs applications, la représentation des objets est sous forme de paires attribut-valeur. Les valeurs sont regroupées en catégories et le nom de l'attribut décrit le rôle de la catégorie dans la description de l'objet. Cette représentation se retrouve notamment dans les bases de données traditionnelles. On peut facilement étendre les définitions précédentes à ce type de représentation [Godin 1986]. Une façon triviale de faire l'extension est de considérer chaque paire attribut-valeur comme un élément atomique et de réutiliser les mêmes définitions.

Pour illustrer cette idée, le Tableau 2 montre un exemple de représentation avec trois attributs. Cet exemple est inspiré de l'approche de représentation multi-facettes de [Prieto-Diaz 1991a]. Les cinq objets sont des composantes logicielles et les trois attributs correspondent aux facettes dans la terminologie de [Prieto-Diaz 1991a]. Dans ce schéma, chaque composante est perçue comme une *Fonction* opérant sur un *Objet* à l'intérieur d'un *Médium*. Par exemple, la composante #1 effectue un *tri-quick* sur des nombres réels à l'intérieur d'un tableau. Pour se ramener aux définitions précédentes, il s'agit tout simplement de voir chaque couple attribut-valeur comme un élément de E' et de transposer le Tableau 2 en une relation binaire. La Figure 6 montre le treillis de Galois correspondant au Tableau 2. La notation $A:v$ représente un élément de E' où A est la première lettre de l'attribut et v est la valeur.

2.5 *Extension à l'utilisation de connaissances taxinomiques.*

La représentation multi-facettes inclue aussi la possibilité de définir des relations taxinomiques entre les valeurs. La Figure 7 est un exemple de relations taxinomiques pour l'exemple du Tableau 2. On représente le fait que, pour l'attribut *Fonction*, les valeurs *tri-quick* et *tri-heap* sont toutes deux des cas particuliers d'une fonction plus générale: *trier*. Un treillis enrichi peut être construit en tenant compte de ces relations taxinomiques. Par exemple, dans la Figure 6, aucune correspondance n'a été faite entre *tri-heap* et *tri-quick* alors que les connaissances taxinomiques peuvent servir à

inférer le fait que les deux composantes font un tri. La Figure 8 montre le treillis enrichi en tenant compte de ces connaissances. En général, l'utilisation des relations taxinomiques fait apparaître de nouveaux couples par rapport au treillis de Galois. Par exemple, dans la Figure 8, se retrouve le nouveau couple ({1,2}, {F:trier, M:tableau}) qui explicite le fait que les composantes #1 et #2 font toutes deux le tri d'un tableau.

Tableau 2. *Représentation multi-facettes de composantes logicielles.*

	Fonction	Objet	Médium
1	tri-quick	réel	tableau
2	tri-heap	chaîne	tableau
3	tri-fusion	chaîne	fichier
4	imprimer	chaîne	fichier
5	chercher	entier	tableau

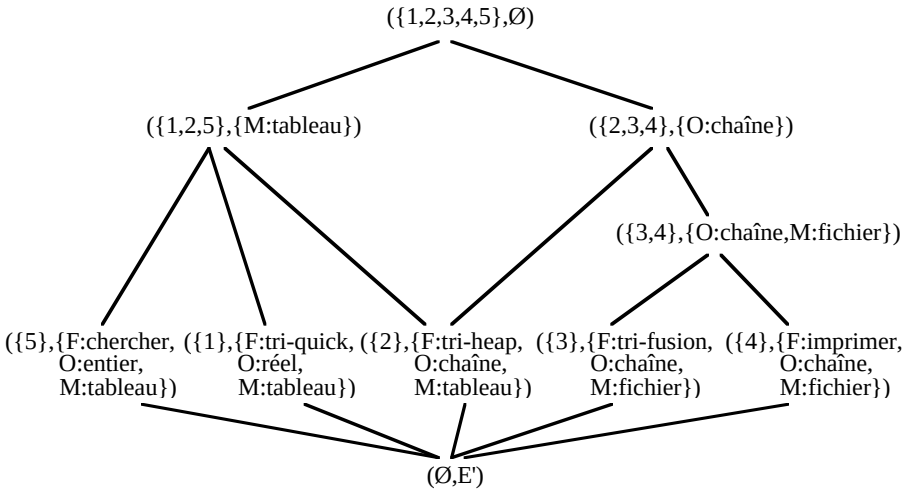

Figure 6. *Treillis de Galois pour la représentation du Tableau 2.*

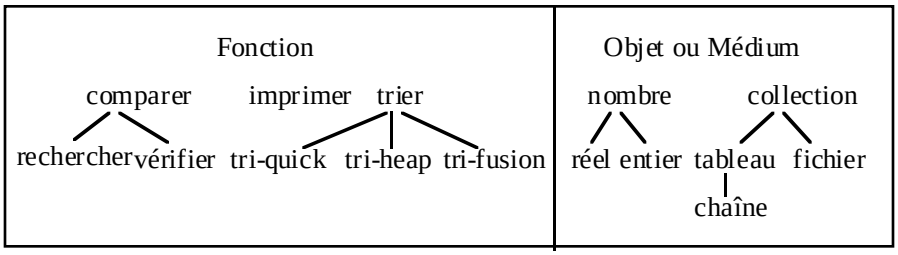

Figure 7. *Taxinomie pour les données du Tableau 2.*

Les définitions des sections précédentes sont facilement étendues en augmentant la relation R de façon à tenir compte des relations taxinomiques. La relation taxinomique est une relation d'ordre partiel notée "<". Par exemple, nous avons que *Fonction:quicksort* < *Fonction:tri*. Pour tenir compte de cette relation taxinomique, définissons une nouvelle relation R^+ entre les éléments de E et de E' :

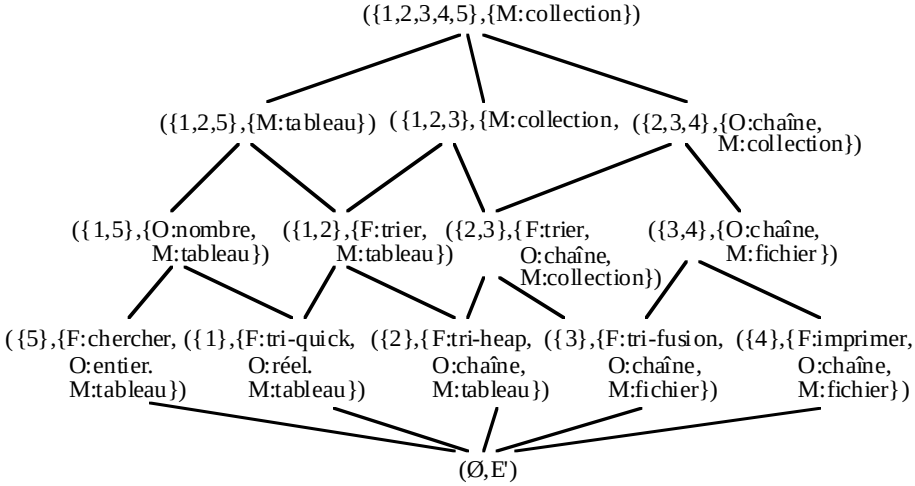

Figure 8. *Treillis de Galois enrichi des connaissances taxinomiques ($G_<$).*

xR^+x' si $\exists y'$ tel que $y' \leq x'$ et xRy'.

Dans l'exemple du Tableau 2, on obtient que *1* R^+ *Fonction:tri-quick* et en plus, la relation *1* R^+ *Fonction:trier*. tient à cause de la relation taxinomique *tri-quick* <*trier*. Pour adapter les définitions de la section 2.2 au cas présent, on n'a qu'à remplacer R par R^+. Un couple (X,X') est complet$_<$ si :

1) $X' = f(X)$ où $f(X) = \{x' \in E' \mid \forall x \in X, xR^+x'\}$
2) $X = f'(X')$ où $f'(X') = \{x \in E \mid \forall x' \in X', xR^+x'\}$.

Par exemple, le couple ({1,2},{F:tri, M:tableau, M:collection}) est complet$_<$. À noter que lorsque deux éléments de E ont une valeur en commun, ils ont nécessairement tous les ancêtres dans la taxinomie pour cette valeur en commun. Dans l'exemple, comme les deux composantes ont en commun *M:tableau*, le parent de *tableau*, i.e *collection*, apparaît aussi dans le couple complet. Dans la plupart des applications, il est naturel de considérer l'élimination de cette information redondante et de ne conserver que la description la plus spécifique. Dans ce cas, il faut modifier un peu la définition pour éviter cette redondance. Un couple est complet$_<$ si :

1) $X' = f(X)$ où $f(X) = \{x' \in E' \mid \forall x \in X, xR^+x'$ et $\neg\exists y'$ tel que $y'<x'$ et $\forall x \in X, xR^+y'\}$.

2) $X = f'(X')$ où $f'(X') = \{x \in E \mid \forall x' \in X', xR^{+}x'\}$.

Nous adopterons plutôt cette dernière définition dans ce qui suit. Le treillis de Galois incorporant la relation taxinomique, noté $G_{<}$, est l'ensemble de ces couples complets. Le treillis résultant apparaît à la Figure 8.

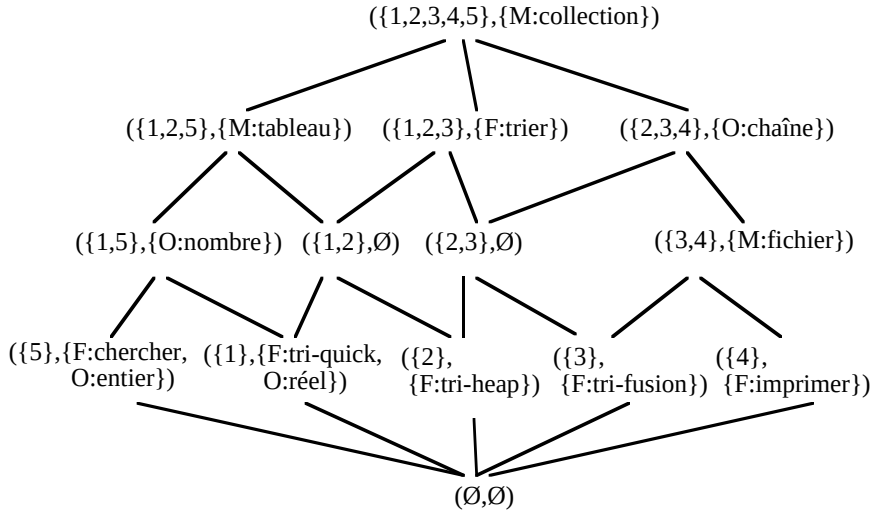

Figure 9. *Un treillis d'héritage selon X' pour la relation taxinomique < ($G_{<}$-HX').*

On peut aussi adapter les autres variations de treillis avec héritage et avec élagage de façon analogue en réutilisant les nouvelles définitions de f et f'. Un treillis d'héritage selon X' pour la relation taxinomique < ($G_{<}$-HX') est l'ensemble des couples (X,$r(X')$) tel qu'il existe un couple complet$_{<}$ (X,X') et :

$r(X') = \{x' \in E' \mid f'(\{x'\}) = X$ et $\neg\exists y' \in E'$ tel que $y'<x'$ et $f'(\{y'\}) = X\}$.

La Figure 9 montre l'exemple du treillis $G_{<}$-HX' pour le Tableau 2 et la taxinomie de la Figure 7.

Un treillis d'héritage ($G_{<}$-H) est défini comme l'ensemble des couples ($r(X)$,$r(X')$) tel qu'il existe un couple complet (X,X') et :

1) $r(X) = \{x \in E \mid f(\{x\}) = X'\}$
2) $r(X') = \{x' \in E' \mid f'(\{x'\}) = X$ et $\neg\exists y' \in E'$ tel que $y'<x'$ et $f'(\{y'\}) = X\}$.

Le treillis d'héritage selon X' élagué selon X' ($G_{<}$-HX'/X') est l'ensemble des couples (X,$r(X')$) satisfaisant :

1) $X = f'(r(X'))$
2) $r(X') = \{x' \in E' \mid f'(\{x'\}) = X$ et $\neg\exists y' \in E'$ tel que $y'<x'$ et $f'(\{y'\}) = X\}$.

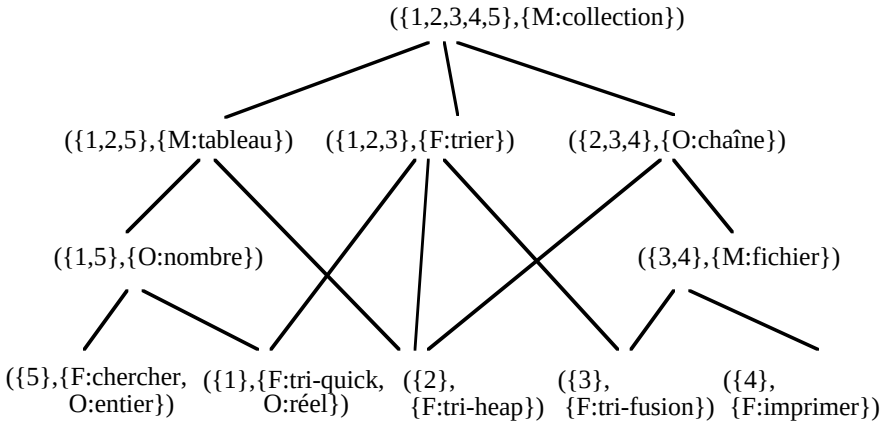

Figure 10. *Le treillis d'héritage selon X' élagué selon X' pour la relation taxinomique* $< (G_{<}\text{-}HX'/X')$.

L'exemple pour les données du Tableau 2 et la taxinomie de la Figure 7 apparaît à la Figure 10. Un aspect important découlant des définitions précédentes est qu'il n'est pas nécessaire d'utiliser exactement les mêmes attributs pour tous les éléments. Ceci est avantageux lorsque l'on veut inclure des objets de diverses natures qui n'ont pas tous la même structure. De plus, il n'est pas obligatoire de restreindre la représentation d'un objet à une valeur par attribut.

L'impact de l'utilisation d'une taxinomie sur la complexité de ces structures est attribuable au fait que la nouvelle fonction $f(X)$ définie à partir de R^+ ajoute les ancêtres des éléments auxquels X est relié. Dans le pire cas, $|f(X)| \leq ka$ où k est comme précédemment la borne supérieure sur le nombre d'éléments reliés à un élément de E et a est une borne supérieure sur le nombre d'ancêtres dans la taxinomie. Le treillis $G_{<}$ est donc borné par :

$$|G_{<}| \leq 2^{ka}\,|E|.$$

En transposant l'analyse empirique sur le treillis de Galois, on obtient que :

$$|G_{<}| \leq k'\,a'\,|E|,$$

où k' et a' sont les valeurs moyennes correspondant à k et a. Pour le treillis élagué:
$|G_{<}\text{-}HX'/X'| \leq ka\,|E|$.

Par analogie avec $G\text{-}HX'/X'$, on peut aussi espérer une croissance logarithmique.

3. Applications

Cette section traite de l'utilisation des structures définies précédemment dans diverses applications.

3.1. *Recherche documentaire*

La recherche booléenne est typique de la plupart des systèmes actuels de recherche documentaire interactive [Salton 1989]. Chaque item est représenté par une liste de termes d'index utilisés dans la formulation des requêtes. Les capacités de formulation de requête booléenne sont bien adaptées au contexte où l'utilisateur a une idée précise de ce qu'il cherche et connaît bien le langage d'indexation du système. Dans le cas contraire, une méthode de repérage axée sur le butinage ("browsing") serait avantageuse [Thompson 1989]. Le butinage est basé sur l'exploration libre d'une structure telle qu'un arbre ou un graphe. Cette forme d'interaction tire profit d'une caractéristique importante de la cognition humaine: il est plus facile de reconnaître quelque chose d'intéressant que de le décrire.

La navigation dans une classification hiérarchique est une forme répandue de méthode de recherche axée sur le butinage. La plupart des systèmes de répertoire de fichiers sont basés sur une organisation hiérarchique de documents et de sous-répertoires. L'utilisateur effectue sa recherche en naviguant interactivement dans cette structure. Un problème important de cette approche est la difficulté de maintenir manuellement la classification. Un autre problème est la rigidité de la classification où un seul chemin d'accès n'est possible pour retrouver un item à partir du point d'entrée. Avec l'ambiguïté des descripteurs de classes [Furnas 1983], une mauvaise décision dans le parcours devient fatale. Un paradigme plus récent axé sur le butinage est l'approche hypertexte [Nielsen 1990]. On retrouve le même problème de maintenance manuelle de la structure. La souplesse supplémentaire de la structure est souvent au prix d'une plus grande difficulté conceptuelle dans le processus navigation.

Reconnaissant les capacités complémentaires de ces approches, plusieurs systèmes hybrides offrent la possibilité d'utiliser plusieurs modes d'interaction différents. Du point de vue du système, un inconvénient majeur est la nécessité de maintenir ces divers appareillages. Pour l'utilisateur, une seule méthode peut être utilisée à la fois et il est difficile de sauter d'une méthode à l'autre parce qu'elles sont basées sur des espaces de recherche indépendants.

L'utilisation du treillis de Galois généré à partir d'une relation d'indexage est une alternative attrayante parce que deux modes d'interaction peuvent être combinés dans un système intégré et cohérent à partir du même espace de recherche. Chaque classe du treillis correspond à un ensemble de documents décrits par les termes d'index communs. Dans la perspective de la recherche booléenne, chaque classe peut être vue comme une requête formée de la conjonction des termes d'index de la classe. Le graphe représente une relation de généralisation/spécialisation entre les requêtes. La recherche est effectuée par une combinaison libre de (1) la spécification directe de termes d'index, résultant en un saut dans la classe la plus générale incorporant les termes spécifiés et les termes de la classe de départ, et (2) la navigation libre en suivant les arcs du graphe du treillis. Le parcours d'un arc correspond à un élargissement (généralisation) ou un raffinement (spécialisation) minimal par rapport

à la requête correspondant à la classe courante. Le second mode d'interaction offre une solution au problème de raffinement de requête des systèmes booléens. Une interface utilisateur permettant de combiner les deux modes d'interaction a été développée sur plate-forme Macintosh™ à l'aide des outils de développement d'interface. Des expériences contrôlées ont mis en évidence le potentiel de cette méthode par rapport à la recherche booléenne et la navigation dans une classification hiérarchique pour le problème de recherche d'un sujet particulier dans le contexte d'un système public [Godin 1993] et d'un système personnel.[Godin 1993]. Les expériences contrôlées ont porté sur un ensemble de 113 résumés tirés du répertoire des films et vidéos de l'Office National du Film du Canada. Pour l'expérience effectuée dans le contexte d'un système public [Godin 1993], le nombre moyen de termes d'index par document est de 6,5 et le treillis résultant contient 325 éléments (paires complètes). Une autre expérience a été effectuée sur un corpus de 3042 rapports techniques indexés par 11,1 termes en moyenne. Le treillis résultant a une taille de 23,471 éléments [Godin 1986]. Ces données impliquent qu'il n'est évidemment pas possible de visualiser directement toute la structure dans le processus de navigation. Une approche simple qui s'est avérée suffisamment efficace est de montrer les prédécesseurs et successeurs immédiats par rapport au contexte courant de navigation.

Une question importante pour l'applicabilité de cette approche est la complexité de la structure et des algorithmes de construction du treillis. La section 2.1 traite de la complexité de la structure. Moyennant une hypothèse raisonnable dans le contexte de la recherche documentaire, le treillis croît linéairement par rapport au nombre de documents. D'autre part, plusieurs algorithmes ont été conçus pour générer le treillis de Galois [Bordat 1986, Carpineto 1993, Chein 1969, Fay 1975, Ganter 1984, Godin 1995, Malgrange 1962, Norris 1978] . Parmi ceux-ci, des algorithmes incrémentaux [Carpineto 1993, Godin 1995] permettent de mettre à jour et le treillis et le graphe. Cette caractéristique est importante en recherche documentaire où de nouveaux documents sont fréquemment ajoutés à la collection. Nous avons expérimenté plusieurs implémentations d'algorithmes incrémentaux et des données empiriques provenant de plusieurs applications ont démontré que l'adjonction d'un nouveau document est faite en temps linéaire par rapport au nombre de documents [Godin 1991, Godin 1993]. Avec l'hypothèse d'une borne supérieure sur le nombre de termes d'index par document, l'analyse dans le pire des cas montre aussi une complexité linéaire. Un résultat surprenant est qu'une variante simple d'un algorithme incrémental s'est avérée meilleure que les algorithmes non incrémentaux testés dans la plupart des cas lorsque l'on cumule le temps pour mettre à jour incrémentalement le treillis [Godin 1995].

Dans la plupart des implantations, nous avons choisi la variante du treillis d'héritage selon X (G-HX). On peut cependant envisager d'autres variantes en fonction des contraintes de l'application. Notre choix a été guidé par le fait que la partie X est plus rarement consultée par l'utilisateur et la consusltation se fait habituellement dans le cas de cardinalités restreintes de X. Il est donc peu coûteux de reconstruire l'ensemble X à partir des $r(X)$ de G-HX en cas de besoin. Par contre il nous semblait qu'il était important pour l'utilisateur de connaître la requête courante, i.e. X', dans sa totalité. Par ailleurs, comme le nombre de termes communs à un ensemble de documents est de taille réduite, la taille de X' est très raisonnable pour envisager sa conservation. Cependant, des expériences récentes nous font croire que le treillis

d'héritage selon X' serait dans certains cas intéressant même pour l'utilisateur en faisant ressortir explicitement les différences entre les classes. Dans une implantation plus récente, l'utilisateur peut alterner entre les deux représentations.

Un autre type d'application de repérage de l'information a été effectué sur un corpus de 101 images qui correspondent à des peintures du National Gallery of Arts de Washington. Chaque image est représentée par un graphe conceptuel. La structure d'espace de connaissance (voir section 3.5) générée à partir des graphes conceptuels a été proposée comme médium de repérage [Mineau 1992]. Un prototype de système de repérage a été réalisé en C sur plate-forme Amiga™. Les fonctions vidéo de l'Amiga ont été utilisées pour l'affichage des images à partir d'un vidéodisque. L'interface permet à l'utilisateur de naviguer dans l'espace de connaissance. Les images d'une classe peuvent être affichées une par une. L'algorithme implanté fonctionne en lot et passe par la génération du treillis d'héritage selon X' élagué selon X' à partir de la représentation en triplets tel que décrit à la section 3.5. Le vocabulaire conceptuel utilisé contient 400 concepts, 80 relations et 300 individus (concepts correspondants à des individus représentés par des noms propres). La structure obtenue contient 591 concepts.

3.2 *Réutilisation*

La réutilisation dans le contexte du développement de logiciel est la création de logiciels à partir d'artefacts existants afin d'augmenter la productivité et la qualité du logiciel. Faute de pouvoir automatiser le processus de développement de logiciels, réutiliser les composantes logicielles d'une application à une autre est une alternative à considérer pour améliorer la productivité et la fiabilité des logiciels. Du point de vue méthodologique, la recherche en réutilisation de logiciels tente de résoudre deux problèmes complémentaires [Krueger 1992] : 1) comment construire des artefacts plus réutilisables, et 2) comment concevoir des logiciels à partir d'artefacts réutilisables. Les artefacts réutilisés peuvent être de diverses natures comme des spécifications, des fragments de code source, des structures de conception, des modèles de données, des données de gestion du processus et autres [Krueger 1992]. Dans un vaste projet de développement d'outils pour un atelier de génie du logiciel, le projet MACROSCOPE [Godin 1995], nous avons utilisé diverses méthodes de classification conceptuelle pour le repérage et l'abstraction de nouveaux artefacts plus réutilisables.

Plusieurs chercheurs ont proposé l'utilisation de mécanismes de repérage basé sur le butinage dans le contexte de la réutilisation parce qu'il n'y a pas toujours d'appariement exact entre la requête et la description de l'artefact, soit à cause de variations dans le vocabulaire, soit parce qu'aucun artefact ne correspond exactement au besoin. Dans ce cas, l'utilisateur voudra explorer le voisinage pour un artefact suffisamment proche de son besoin. La classification conceptuelle peut servir au butinage et à la compréhension des relations entre les artefacts du répertoire.

Dans une expérience de repérage, nous avons utilisé *G-HX* et *G-H/X'* pour le repérage d'éléments de données provenant du dictionnaire de données de la Banque Nationale du Canada. Le dictionnaire sert à uniformiser le langage utilisé dans le processus de développement des applications de la banque. Les analystes doivent

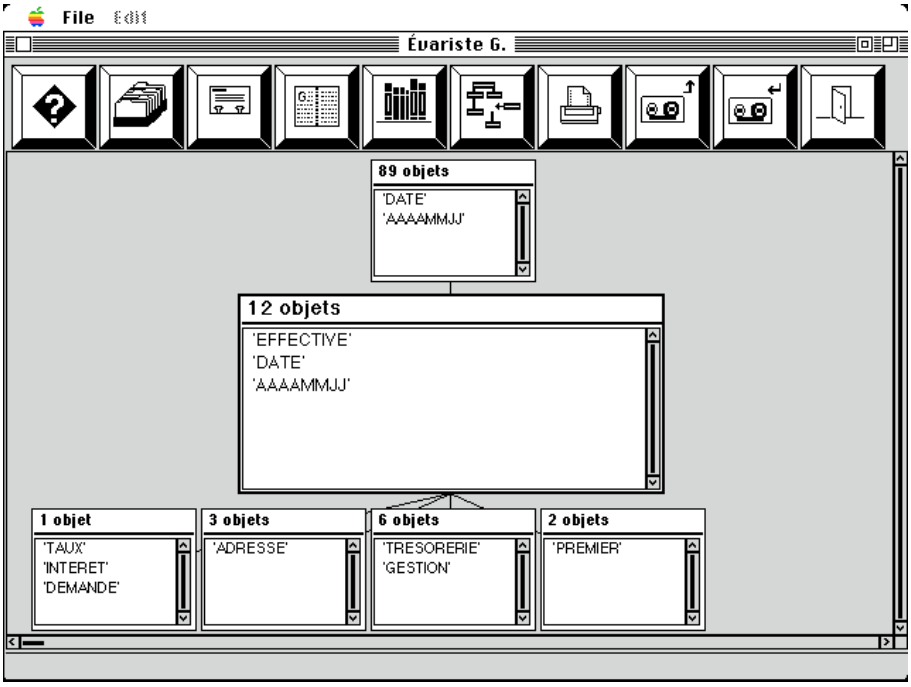

Figure 11. *Exemple d'écran pour l'application du reprérage d'éléments de données.*

consulter le dictionnaire pour déterminer si des éléments existants peuvent satisfaire aux besoins d'une nouvelle application. Étant donné le grand volume du dictionnaire, la recherche est un processus difficile. Pour assister la tâche, les éléments sont indexés manuellement. Les méthodes de classification ont été appliquées à la relation d'indexage pour le repérage. Un prototype a été développé dans l'environnement Visualworks™. La portabilité de cet environnement a permis de porter le prototype sur diverses plates-formes. Le prototype permet de générer incrémentalement les diverses combinaisons de treillis avec héritage et élagage. Pour le treillis élagué, *G-H/X'*, un algorithme incrémental a été implanté et des analyses de performance ont démontré que les variantes élaguées peuvent être générées beaucoup plus efficacement que le treillis complet principalement à cause de la complexité moindre [Godin 1995]. Le principe de l'interface est le même que pour l'application de recherche documentaire décrite à la sous-section précédente. Les avantages respectifs de chaque structure sont mis en relief dans [Godin 1995].

Dans une expérience, les hiérarchies ont été générées pour un sous-ensemble du répertoire contenant 796 éléments de données décrits par 5,4 termes en moyenne. Le treillis *G-HX* contient 1868 éléments alors que le treillis élagué en contient 639. La Figure 11 montre une copie d'écran de l'interface utilisateur pour *G-HX*. La requête courante dans la navigation correspond à la paire contenant l'ensemble de termes {'EFFECTIVE', 'DATE' ,'AAAAMMJJ'} et 12 éléments (objets) du répertoire. L'utilisateur peut au besoin demander l'affichage de la liste des éléments de la requête

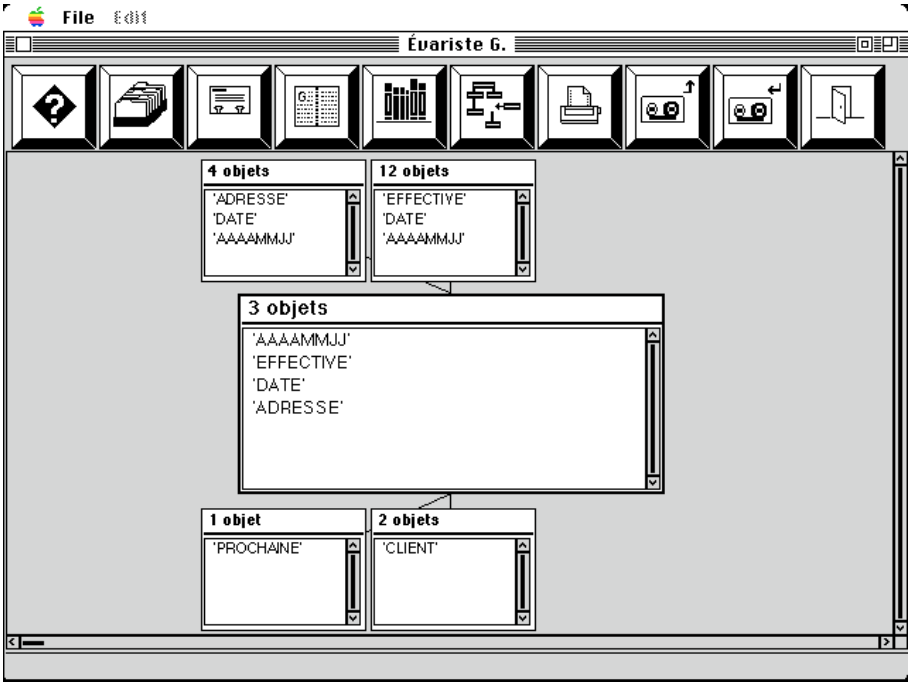

Figure 12. *Écran résultant de la sélection du successeur contenant 'ADRESSE' dans l'écran précédent.*

courante. L'utilisateur pourrait par exemple aboutir à cet état en recherchant un élément de données correspondant à l'adresse effective du changement d'adresse d'un client. Il aurait pu aboutir à cet état soit en spécifiant directement des termes d'index ou en naviguant à partir du haut du treillis (inf(*G*)). Dans cet exemple, s'il avait spécifié DATE et EFFECTIVE, il aurait abouti à cet état qui correspond à la paire la plus générale contenant les deux termes. Ceci est déductible par le fait qu'aucun des prédécesseurs ne contient ces deux termes. D'ailleurs, si l'utilisateur n'avait spécifié que le terme 'EFFECTIVE', l'effet aurait été le même. Il y a un prédecesseur et quatre successeurs immédiats dans le treillis qui peuvent être sélectionnés par l'utilisateur pour poursuivre la navigation. Notons que pour éviter la redondance, les termes de la paire courante ne sont pas répétés dans les successeurs. Un attrait important pour l'utilisateur de cette approche est de se voir offrir des alternatives de raffinement de sa requête qui sont pertinentes par rapport au contenu du répertoire. Dans notre exemple, l'utilisateur serait amené à sélectionner le successeur contenant le terme 'ADRESSE'. Ceci aurait pour effet de changer l'élément courant dans la navigation et d'afficher l'écran de la Figure 12.

Afin de donner une idée plus concrète du coût de la génération de ces structures pour un grand volume de données, la Figure 13 montre le résultat d'une expérience récente où apparaît le temps total nécessaire pour générer incrémentalement les deux structures pour le répertoire complet contenant 4675 éléments. Les éléments sont

ajoutés un à un et le temps total est la somme des temps d'adjonction de tous les éléments. La figure montre l'évolution du temps total avec le nombre d'éléments. On peut vérifier que le treillis est plus coûteux à générer que le treillis élagué et que la différence s'accroît avec la taille du répertoire. La génération totale peut devenir assez coûteuse pour un ensemble de cette taille, soit 13992 secondes pour G et 4615 secondes pour G-HX'/X'. Par contre, même lorsque l'on se rapproche du nombre total d'éléments, le temps nécessaire pour ajouter un nouvel élément demeure raisonnable, soit de l'ordre de 6 secondes pour le treillis et 2 secondes pour le treillis élagué. Ces expériences ont été effectuées sur une station SUN SPARC IPC.

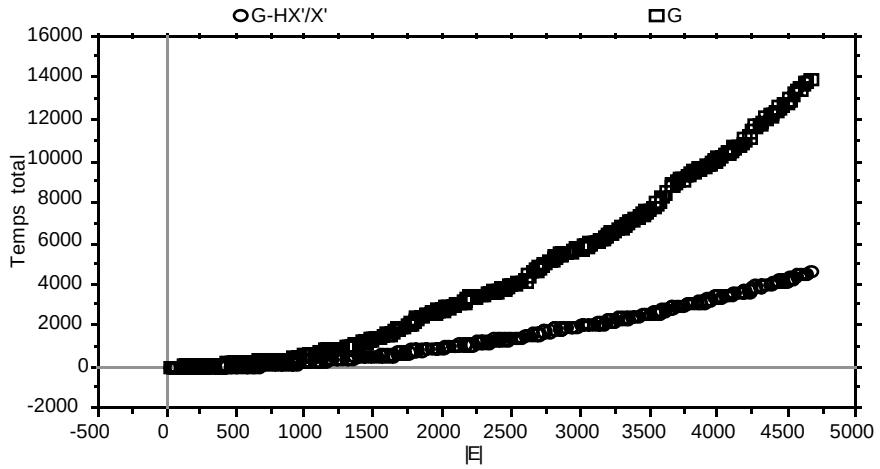

Figure 13. *Temps total pour la génération de G et G-HX'/X'.*

Une autre expérience de repérage a été conduite sur le texte de la méthode de développement de DMR, Productivité Plus™. Le texte a été indexé automatiquement en utilisant une méthode d'indexation basée sur une analyse syntaxique sophistiquée [Faraj 1994]. La relation d'indexation a servi à produire les hiérarchies pour le repérage.

Nous avons également implanté les extensions permettant l'utilisation d'attributs et de taxinomies tel que décrits à la section précédente. Les algorithmes incrémentaux de mise à jour ont été adaptés à cet effet. Un expérience est en cours sur la recherche de composantes décrites avec une représentation multi-facettes telle que proposée par [Prieto-Diaz 1991a]

Les méthodes de classification peuvent aussi servir à identifier de nouveaux artefacts avec un meilleur potentiel de réutilisation. Un premier effort dans cette direction a porté sur un répertoire de définition d'entités en termes de leurs attributs. Les attributs font référence aux éléments de données décrits précédemment. Les entités du répertoire ne font pas ressortir les parties communes de façon explicite comme on retrouve dans les modèles sémantiques ou objets. La génération automatique des hiérarchies à partir de la relation entité-attribut a permis d'identifier des entités génériques à partir des entités existantes. Ces nouvelles entités ont un meilleur potentiel de réutilisation. Les variantes *G-H* et *G-H/X'* ont été utilisées pour faire

ressortir de façon explicite les différences entre les entités génériques et les spécialisations en termes des attributs à la manière des hiérarchies d'héritage dans les modèles sémantiques ou objets. Les avantages relatifs de chaque structure sont aussi discutés dans [Godin 1995].

Une approche plus ambitieuse à la réutilisation est d'identifier des familles larges de composantes reliées entre elles qui peuvent être extraites et réutilisées en bloc. À cet effet, nous avons proposé d'appliquer une structure dérivée de la notion de treillis d'héritage, appelée espace de connaissance (voir section 3.5), pour induire une hiérarchie de modèles de données génériques à partir de modèles existants d'un domaine d'application [Mineau 1993]. La méthode en soi n'est pas limitée aux modèles de données et peut être appliquée à d'autres produits du processus de développement de logiciel qui sont représentés sous forme de graphes. L'objectif est de pouvoir découvrir les sous-modèles communs du domaine d'application qui sont d'un haut niveau de réutilisabilité.

3.3 Conception des hiérarchies de classes dans le développement orienté objet

Une étape centrale dans la modélisation orientée objet consiste à identifier les objets d'une application, et les relations sémantiques les reliant. Les relations d'héritage entre les différentes classes ont une influence majeure sur : 1) la compréhensibilité des modèles, 2) l'intégrité sémantique des modèles, 3) le potentiel de réutilisation des classes résultantes pour d'autres applications, et 4) l'effort de développement requis. Plusieurs chercheurs ont remarqué que certaines hiérarchies de classes ne sont pas naturelles [Cook 1992, Cox 1990], et sont souvent sous-optimales du point de vue de la factorisation des propriétés communes entre classes. Le treillis d'héritage et du treillis d'héritage élagué peuvent servir comme base pour des outils de génération et de réorganisation des hiérarchies classes à partir de la spécification de leurs propriétés [Dicky 1994, Godin 1993]. Cette application est une extension naturelle de l'application décrite en 3.2 sur la découverte d'entités génériques à partir de leurs attributs. Deux aspects des structures de treillis d'héritage en font des structures bien adaptées à ce problème. Premièrement, la factorisation des propriétés est maximale, ce qui produit des hiérarchies sans redondance. Deuxièmement, la hiérarchie est conforme à la relation de sous-typage. Ces deux caractéristiques sont importantes pour la conception des hiérarchies de classes [Casais 1991, Johnson 1988, Korson 1992, Lieberherr 1991]. Cette approche se distingue des autres algorithmes proposés pour le même problème par le fait que :

- la hiérarchie peut être construite incrémentalement contrairement à [Dvorak 1994, Lieberherr 1991] ;
- la hiérarchie résultante est bien définie et ne dépend pas de spécificités algorithmiques; ce qui n'est pas le cas pour [Bergstein 1991, Casais 1991, Dvorak 1994, Lieberherr 1991] ;
- la hiérarchie n'est pas limitée à une structure d'arbre contrairement à [Bergstein 1991, Dvorak 1994] ;

- la factorisation des propriétés est maximale, ce qui n'est pas nécessairement le cas pour [Casais 1991, Dvorak 1994].

Les auteurs de [Dicky 1994] ont récemment proposé pour la conception des hiérarchies de classes un algorithme incrémental qui peut aussi produire le treillis d'héritage élagué. Leur algorithme peut aussi s'appliquer à d'autres structures qui respectent la propriété de factorisation maximale.

La méthode de classification utilisée dépend entre autres de la nature des spécifications des propriétés des classes. Une approche simple consiste à ne considérer que la relation binaire définie par le protocole d'interface constitué de l'ensemble de noms des messages auxquels la classe peut répondre. Cook [Cook 1992] a utilisé pour l'analyse des classes Collection de Smalltalk-80™ une structure identique au treillis d'héritage élagué produit à partir du protocole d'interface des classes. La hiérarchie produite, appelée hiérarchie d'interface, a été analysée, ce qui a conduit à la découverte d'inconsistances et à la suggestion d'améliorations à la librairie existante.

Les algorithmes que nous avons développés permettent de générer la hiérarchie de façon efficace et incrémentale si nécessaire. Dans le cadre du projet IGLOO (InGénierie du Logiciel Orienté Objet), nous avons développé un outil qui permet d'appliquer les diverses méthodes de classification au problème de conception des hiérarchies de classes en fonction des propriétés spécifiées [Godin 1994]. Diverses métriques peuvent être calculées pour évaluer les hiérarchies obtenues. L'outil peut aussi extraire l'interface d'un ensemble de classes de l'environnement Smalltalk-80™ et produire la hiérarchie d'interface à partir des algorithmes de génération du treillis et du treillis élagué [Arfi 1994].

Une expérience dans le contexte du projet IGLOO a porté sur la conception d'un modèle objet pour la gestion des réseaux. Les hiérarchies obtenues ont permis de mettre en évidence un certain nombre de classes abstraites et de relations hiérarchiques qui n'avaient pas été identifiées dans le modèle initial.

Des expériences préliminaires effectuées sur la librairie de Smalltalk-80™ [Godin 1994] mettent en lumière les avantages respectifs des hiérarchies produites par le treillis de Galois et le treillis élagué par rapport à la librairie actuelle. La hiérarchie d'interface est souvent plus proche des besoins de l'utilisateur que la hiérarchie d'implantation et peut être utilisée simplement comme médium de recherche pour la réutilisation. Dans une optique semblable, une structure très proche du treillis de Galois a été proposée pour le repérage de classes dans de larges librairies à partir de leurs propriétés [Oosthuizen 1992]. La structure proposée se distingue par le fait que chaque propriété apparaît individuellement au premier niveau, ce qui n'est pas nécessairement le cas dans un treillis de Galois.

La classification peut-être opérée sur une représentation plus riche des propriétés des classes que simplement le protocole d'interface [Dvorak 1994]. Selon la nature de la représentation, les diverses extensions présentées dans ce papier peuvent servir à générer une hiérarchie qui tire profit de la richesse supplémentaire dans la description. Par exemple, pour distinguer les différentes implantations du même message, une relation taxinomique peut être créée entre la spécification abstraite du message, i.e. son interface, et les diverses méthodes qui l'implantent, chaque méthode étant vue comme une spécialisation de la spécification abstraite [Godin 1993]. En appliquant cette idée, on obtient une hiérarchie sans redondance où la spécification de chaque interface et de chaque méthode n'apparaît qu'une seule fois. Des expériences sont en

cours sur la librairie de Smalltalk-80™. Des résultats préliminaires suggèrent que dans certains cas, la richesse supplémentaire du treillis de Galois par rapport au treillis élagué serait beaucoup trop lourde pour être utilisée en pratique. Le nombre de classes abstraites générées peut devenir beaucoup trop grand à cause d'un grand nombre de combinaisons possibles entre les interfaces et les implantations des méthodes.

3.4 Découverte de règles dans les bases de données

On remarque un intérêt grandissant pour les méthodes de découverte de connaissances à partir des données. Ceci est du entre autres au volume croissant des données accumulées par les organisations qui sont largement sous-exploitées. Le treillis de Galois est une représentation efficace pour l'extraction de règles d'implication entre les propriétés d'une relation [Guigues 1986, Wille 1992]. Par exemple, à partir du treillis de la Figure 1, on peut facilement déduire que *e->b,g* parce que le couple le plus général contenant *e* contient aussi *b,g*. Cette règle affirme que tout objet possédant la propriété *e* possède aussi les propriétés *b* et *g*. Des algorithmes ont été développés pour générer les règles à partir du treillis. Pour éviter la reconstruction des règles à chaque mise à jour de la base de données, un algorithme permettant de modifier le treillis et les règles correspondantes de façon incrémentale a été proposé [Godin 1994]. L'analyse des algorithmes permet d'entrevoir l'application à des volumes de données suffisamment larges pour être intéressants en pratique.

3.5 Acquisition et organisation des connaissances

La structure d'espace de connaissance a été proposée comme support au processus d'acquisition et d'organisation des connaissances représentées par graphes conceptuels [Mineau 1992, Mineau 1990, Mineau 1994]. Cette structure correspond à un treillis avec héritage selon X' élagué selon X' sur une relation binaire R définie à partir de la représentation par graphes conceptuels et à partir d'une relation taxinomique dérivée des graphes comme le montre ce qui suit. La hiérarchie produite facilite le traitement de diverses tâches telles que répondre à des requêtes sur des ensembles d'objets, rechercher les similarités et les différences entre les objets. Plutôt que de produire la hiérarchie manuellement, les méthodes de classification conceptuelle telles que la génération de l'espace de connaissance peuvent servir à alléger le processus. En même temps, l'espace de connaissance peut servir à améliorer le processus d'acquisition des connaissances en révélant des inconsistances par rapport aux connaissances déjà acquises.

La Figure 14 montre un exemple comprenant 3 objets représentés suivant le formalisme des graphes conceptuels [Sowa 1984]. Les graphes conceptuels représentent les concepts de base avec des boîtes étiquetées; les relations entre les concepts sont exprimées par des ovales étiquetés. Les concepts utilisés peuvent être reliés hiérarchiquement à d'autres dans une taxinomie appelée *hiérarchie des types*.

La structure d'espace de connaissance pour cet exemple apparaît à la Figure 17. Pour retrouver cette structure dans le cadre des définitions des sections précédentes, on

transforme la représentation par graphes en une représentation par triplets de la forme <$concept_1$,*relation*,$concept_2$>, où *relation* correspond à une relation entre le $concept_1$ et le $concept_2$. La représentation par triplets de l'exemple de la Figure 14 est donnée à la Figure 15. On obtient ainsi une relation binaire *R* entre les objets et les triplets, ce qui permet d'appliquer les définitions de treillis de Galois et des variantes à partir de cette relation *R*.

Cependant, ceci limite l'appariement des graphes à des triplets complets. Les objets 2 et 3 n'ont aucun triplet en commun; cependant les deux derniers triplets ont en commun la relation couleur et le concept brun représentant le fait que les deux animaux ont en commun la couleur brun. Pour permettre un tel appariement partiel sur les triplets, on introduit les *patrons de généralisation* qui sont de nouveaux triplets constitués en remplaçant les relations et/ou concepts d'un triplet par une valeur générale indéterminée ∇. À partir des patrons de généralisation, on obtient une taxinomie illustrée à la Figure 16 pour un exemple de triplet. En utilisant cette taxinomie, on pourra inférer que les objets #2 et #3 ont en commun le patron de généralisation <∇, couleur, brun>. Les ensembles d'objets de l'espace de connaissance sont les mêmes que dans un treillis avec héritage selon *X'* élagué selon *X'* ($G_<$-*HX'/X'*) pour la relation *R* définie sur la représentation par triplet et pour la taxinomie définie sur les patrons de généralisation. La description obtenue n'est pas l'ensemble *r*(*X'*) mais une représentation qu'on peut obtenir à partir de *r*(*X'*) avec un peu de travail supplémentaire. Par exemple dans $G_<$-*HX'/X'*, on obtient le couple ({1,2,3}, {<∇,partie,pattes>, <pattes,qté,∇>, <∇,couleur,∇>}. Pour obtenir la description contenue dans l'espace de connaissance, il faut reconstituer les connections entre les triplets. En plus, on conserve les valeurs des triplets complets qui correspondent à ∇ pour chaque objet. Ces informations peuvent être utilisées pour produire une description plus précise en remplaçant le ∇ par une valeur plus spécifique à partir de la hiérarchie des types.

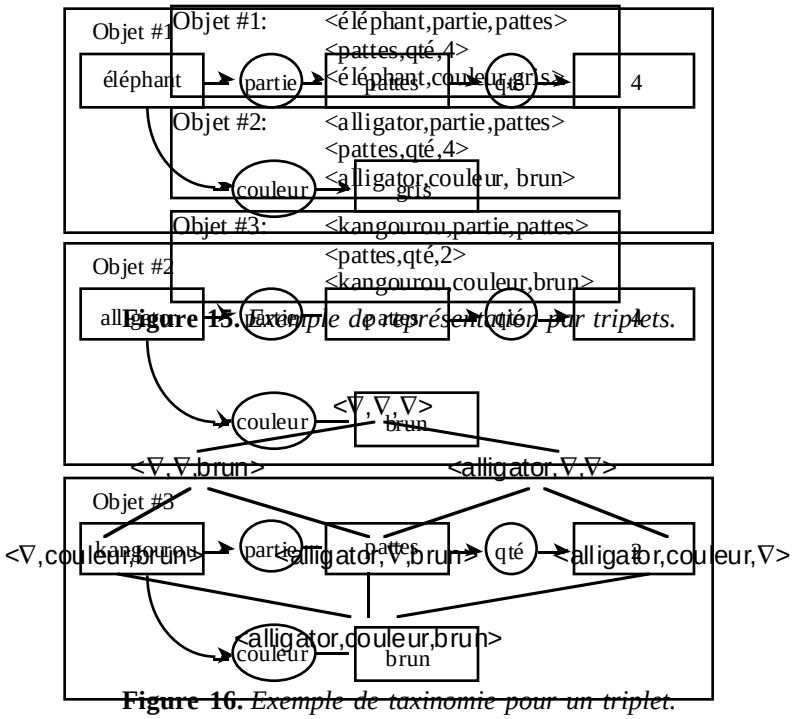

Figure 15. *Exemple de représentation par triplets.*

Figure 16. *Exemple de taxinomie pour un triplet.*

Figure 14. *Exemple de représentation par graphes conceptuels.*

La Figure 18 montre un exemple de hiérarchie des types pour l'exemple de la Figure 14. En se servant de cette taxinomie on peut remplacer les ∇ par le super-type minimal commun lorsqu'il existe, produisant ainsi une description plus révélatrice. Par exemple, pour le couple correspondant à $X = \{1,2\}$, on peut remplacer le ∇ par la valeur *animal*.

À l'aide du cadre conceptuel défini précédemment, on peut considérer toutes les autres alternatives de structures dans le contexte de la représentation par graphes conceptuels. Par exemple, on peut considérer un treillis de Galois sur la représentation par graphes conceptuels. L'impact en terme de complexité par rapport au bénéfice en terme de richesse de la structure est une question importante à considérer.

Les connaissances taxinomiques de la hiérarchie des types ne sont utilisées ici qu'a posteriori pour enrichir la description des couples. On pourrait aussi envisager d'utiliser ces connaissances a priori pour raffiner le processus d'appariement lors de la classification. Dans l'approche présentée précédemment, la taxinomie utilisée pour la classification sur les triplets ne tenait compte que de deux valeurs pour chaque position: la valeur dans le graphe et la valeur ∇. On peut enrichir la taxinomie sur les triplets en tenant compte des valeurs intermédiaires de généralisation contenues dans la hiérarchie des types. Cette approche risque d'introduire une complexité supplémentaire importante si l'on veut considérer toutes les combinaisons possibles

pour chaque position de chaque triplet. Une approche plus judicieuse consiste à ne considérer qu'un ensemble limité de combinaisons possibles par rapport au contexte des triplets dans les graphes conceptuels. Une façon naturelle de limiter les cas possibles en tenant compte du canon de définition des graphes est l'objet de travaux en cours.

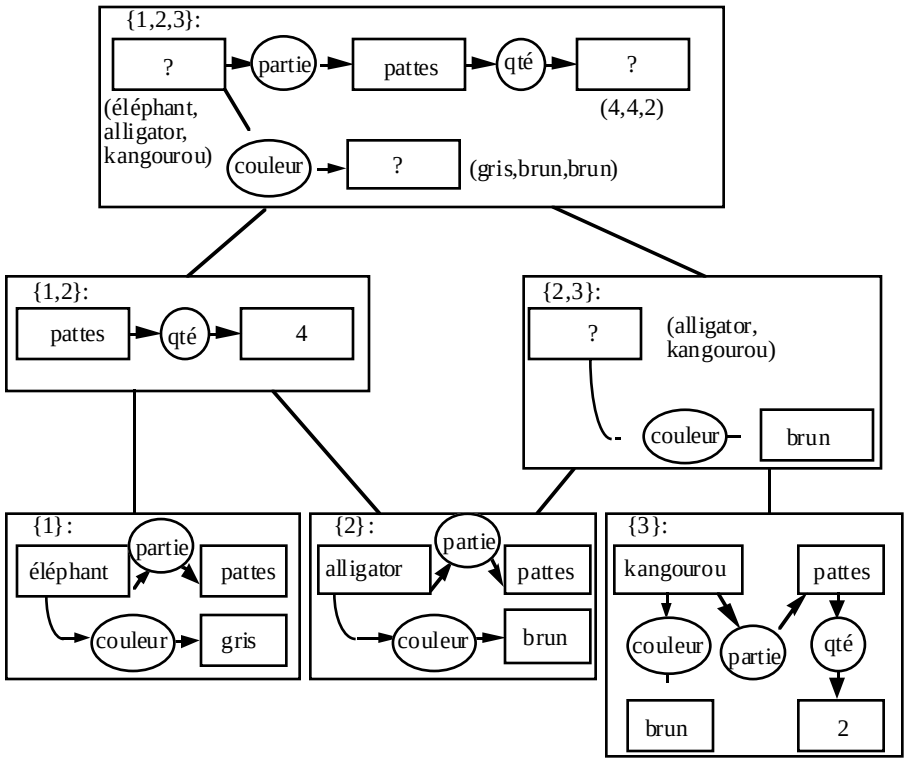

Figure 17. *Espace de connaissance pour l'exemple de la Figure 2.6.1.*

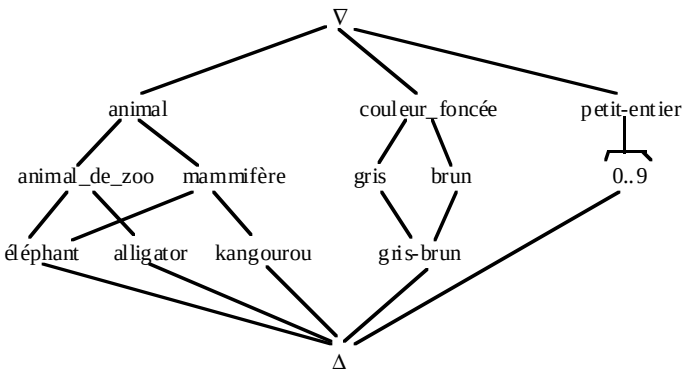

Figure 18. *Hiérarchie des types.*

Conclusion

Diverses structures utilisées dans des méthodes de classification conceptuelle ont été présentées dans un cadre unificateur pour en faire ressortir les points communs. En particulier, ces structures ont été dérivées dans le cadre de la notion de treillis de Galois (treillis de concepts). Cette présentation permet de considérer un grand nombre de structures alternatives en fonction des besoins des applications. L'utilisation de ces structures de classification est illustrée par une variété d'applications: recherche documentaire, réutilisation, conception des hiérarchies de classes, génération de règles d'implication à partir de bases de données, acquisition et organisation des connaissances. L'ampleur, la diversité des applications, le nombre croissant de chercheurs œuvrant dans ce domaine font preuve de la versatilité des notions sous-jacentes.

Remerciements

Ces travaux ont été supportés en partie par des subventions provenant du Conseil National de la Recherche du Canada, du Centre de Recherche en Informatique de Montréal dans le cadre du projet mobilisateur Le Macroscope Informatique géré par le Groupe DMR Inc., et du Ministère de l'Industrie, du Commerce, de la Science et de la Technologie du Québec, pour le projet IGLOO organisé par le Centre de Recherche en Informatique de Montréal.

Bibliographie

[ANQ 94] ANQUETIL, N. J., VAUCHER, J., «Extracting Hierarchical Graphs of Concepts from an Object Set», *Proceedings of the International Conference on Conceptual Structures* (1994), p. 26-45.

[ARF 94] ARFI, A., GODIN, R., MILI, H., MINEAU, G., MISSAOUI, R., «A Tool for the Automatic Generation of the Interface Hierarchy for the Smalltalk-80 Class Library», *Proceedings of the Acfas Colloquium on Object Orientation in Databases and Software Engineering - Colloque Orientation Objet en Bases de Données et Génie du Logiciel* (1994), p. 61-78.

[BAR 70] BARBUT, M., MONJARDET, B., *Ordre et Classification. Algèbre et Combinatoire, Tome II*, Hachette, 1970.

[BER 91] BERGSTEIN, P., LIEBERHERR, K., «Incremental Class Dictionary Learning and Optimization», *Proceedings of the European Conference on Object-Oriented Programming* (1991), p.

[BOR 86] BORDAT, J. P., «Calcul Pratique du Treillis de Galois d'une Correspondance», *Mathématiques et Sciences Humaines* , 96, 1986, p. 31-47.

[BRO 77] BROOKES, B. C., «Theory of the Bradford Law», *Journal of Documentation* , 33, n° 3, 1977, p. 180-209.

[CAR 90] CARBONELL, J. G., Introduction: Paradigms for Machine Learning, in *Machine Learning: Paradigms and Methods*, J. G. Carbonell (Eds.), The MIT Press, p. 1-9, 1990.

[CAR 93] CARPINETO, C., ROMANO, G., «GALOIS: An Order-Theoretic Approach to Conceptual Clustering», *Proceedings of the Machine Learning Conference* (1993), p. 33-40.

[CAS 91] CASAIS, E. , Managing Evolution in Object Oriented Environments: An Algorithmic Approach, thèse de doctorat, Université de Genève, 1991.

[CHE 69] CHEIN, M., «Algorithme de Recherche des Sous-Matrices Premières d'une Matrice», *Bull. Math. Soc. Sci. Math. R.S. Roumanie* , 13, 1969, p. 21-25.

[COO 92] COOK, W. R., «Interfaces and Specifications for the Smalltalk-80 Collection Classes», *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications* (1992), p. 1-15.

[COX 90] COX, B.-J., «Planning the Software Revolution», *IEEE Software* , 7, n° 6, 1990, p. 25-35.

[DAN 93] DANIEL-VATONNE, M. C., DE LA HIGUERA, C., «Les termes: un modèle algébrique de représentation et de structuration des données symboliques», *Mathématique informatique et sciences humaines* , 122, 1993, p. 41-63.

[DAV 92] DAVEY, B. A., PRIESTLEY, H. A., *Introduction to Lattices and Order*, Cambridge University Press, 1992.

[DIC 94] DICKY, H., DONY, C., HUCHARD, M., LIBOUREL, T., «Un algorithme d'insertion avec restructuration dans les hiérarchies de classes», *Proceedings of the Langages et Modèles à Objets* (1994), p.

[DVO 94] DVORAK, J., «Conceptual Entropy and Its Effect on Class Hierarchies», *IEEE Computer* , 27, n° 6, 1994, p. 59-63.

[FAR 94] FARAJ, N., GODIN, R., MISSAOUI, R., DAVID, S., PLANTE, P., «Évaluation de la contribution des termes composés syntaxiques pour l'indexation automatique», *Proceedings of the 22e Congrès Annuel de l'Association Canadienne des Sciences de l'Information* (1994), p. 340-360.

[FAY 75] FAY, G., «An Algorithm for Finite Galois Connexions», *Journal of Computational Linguistic and Languages* , 10, 1975, p. 99-123.

[FIS 87] FISHER, D., «Knowledge Acquisition via Incremental Conceptual Clustering», *Machine Learning* , 2, 1987, p. 139-172.

[FIS 91] FISHER, D. H., PAZZANI, M. J., LANGLEY, P., Eds (1991). *Concept Formation: Knowledge and Experience in Unsupervised Learning*. Morgan Kaufmann, 1991.

[FUR 83] FURNAS, G. W., LANDAUER, T. K., GOMEZ, L. M., DUMAIS, S. T., «Statistical Semantics: Analysis of the Potential Performance Of Key-Word Information Systems», *The Bell System Technical Journal* , 62, 1983, p. 1753-1806.

[GAN 84] GANTER, B., Two Basic Algorithms in Concept Analysis, Preprint 831, Technische Hochschule Darmstadt, 1984.

[GAN 86] GANTER, B., STAHL, J., WILLE, R., Conceptual Measurement and Many-Valued Contexts, in *Classification as a Tool of Research*, W. Gaul, M. Schader (Eds.), North-Holland, p. 169-176, 86.

[GAN 89] GANTER, B., WILLE, R., Conceptual Scaling, in *Applications of Combinatorics and Graph Theory to the Biological and Social Sciences*, F. Roberts (Ed.), Springer-Verlag, p. 139-167, 1989.

[GEN 90] GENNARI, J. H., LANGLEY, P., FISHER, D., Models of Incremental Concept Formation, in *Machine Learning: Paradigms and Methods*, J. Carbonell (Ed.), MIT Press, p. 11-62, 1990.

[GOD 86] GODIN, R., SAUNDERS, E., GECSEI, J., «Lattice Model of Browsable Data Spaces», *Information Sciences* , 40, 1986, p. 89-116.

[GOD 89a] GODIN, R., «Complexité de Structures de Treillis», *Annales des Sciences Mathématiques du Québec* , 13, n° 1, 1989, p. 19-38.

[GOD 89b] GODIN, R., GECSEI, J., PICHET, C., «Design of a Browsing Interface for Information Retrieval», *Proceedings of the 12th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1989), p. 32-39.

[GOD 91] GODIN, R., MISSAOUI, R., ALAOUI, H., «Learning Algorithms Using a Galois Lattice Structure», *Proceedings of the Third International Conference on Tools for Artificial Intelligence* (1991), p. 22-29.

[GOD 93a] GODIN, R., DAVIDSON, M., MISSAOUI, R., MILI, H., «Experimental Comparison of three Methods for Personal Information Retrieval», *Proceedings of the Second Annual Symposium on Document Analysis and Information Retrieval* (1993), p. 379-398.

[GOD 93b] GODIN, R., MILI, H., «Building and Maintaining Analysis-Level Class Hierarchies Using Galois Lattices», *Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'93)* (1993), p. 394-410.

[GOD 93c] GODIN, R., MINEAU, G., MISSAOUI, R. Projet Macroscope: Volet Réutilisation. Phase 3. CRIM/DMR, 1993.

[GOD 93d] GODIN, R., MISSAOUI, R., APRIL, A., «Experimental Comparison of Navigation in a Galois Lattice with Conventional Information Retrieval Methods», *International Journal of Man-Machine Studies* , 38, 1993, p. 747-767.

[GOD 94a] GODIN, R., MILI, H., ARFI, A., MINEAU, G. W., MISSAOUI, R., «A Tool for Building and Evaluating Class Hierarchies Based on a Concept Formation Approach», *Proceedings of the OOPSLA 94 Workshop on Artificial Intelligence for Object-Oriented Software Engineering* (1994).

[GOD 94b] GODIN, R., MISSAOUI, R., «An Incremental Concept Formation Approach for Learning from Databases», *Theoretical Computer Science, Special Issue on Formal Methods in Databases and Software Engineering* , 133, 1994, p. 387-419.

[GOD 95a] GODIN, R., MINEAU, G., MISSAOUI, R., ST-GERMAIN, M., FARAJ, N., «Applying Concept Formation Methods to Software Reuse», *International Journal of Knowledge Engineering and Software Engineering* , 5, n° 1, 1995, p. 119-142.

[GOD 95b] GODIN, R., MISSAOUI, R., ALAOUI, H., «Incremental Concept Formation Algorithms Based on Galois (Concept) Lattices», *Computational Intelligence* , 11, n° 2, 1995, p. 246-267.

[GUÉ 90] GUÉNOCHE, A., «Construction du Treillis de Galois d'une Relation Binaire», *Mathématiques et Sciences Humaines* , 109, 1990, p. 41-53.

[GUÉ 93] GUÉNOCHE, A., VAN MECHELEN, I., Galois Approach to the Induction of Concepts, in *Categories and Concepts : Theoretical Views and Inductive Data Analysis*, I. Van Mechelen, J. Hampton, R. Michalski, P. Theüns (Eds.), Academic Press, p. 287-308, 1993.

[GUI 86] GUIGUES, J. L., DUQUENNE, V., «Familles Minimales d'Implications Informatives Résultant d'un Tableau de Données Binaires», *Mathématiques et Sciences Humaines* , 95, 1986, p. 5-18.

[JOH 88] JOHNSON, R., FOOTE, B., «Designing Reusable Classes», *Journal of Object-Oriented Programming* , June/July, 1988, p. 22-35.

[KOR 92] KORSON, T., MCGREGOR, J. D., «Technical Criteria for the Specification and Evaluation of Object-Oriented Libraries», *Software Engineering Journal* , March, 1992, p. 85-94.

[KRO 94] KRONE, M., SNELTING, G., «On The Inference of Configuration Structures from Source Code», *Proceedings of the 16th International Conference on Software Engineering* (1994), p. 49-57.

[KRU 92] KRUEGER, C. W., «Software Reuse», *ACM Computing Surveys* , 24, n° 2, 1992, p. 131-184.

[LEB 87] LEBOWITZ, M., «Experiments with Incremental Concept Formation: UNIMEM», *Machine Learning* , 2, 1987, p. 103-138.

[LIE 91] LIEBERHERR, K. J., BERGSTEIN, P., SILVA-LEPE, I., «From Objects to Classes: Algorithms for Object-Oriented Design», *Journal of Software Engineering* , 6, n° 4, 1991, p. 205-228.

[LIQ 90] LIQUIÈRE, M., MEPHU NGUIFO, E., «LEGAL (Learning with Galois Lattice): Un Système d'Apprentissage de Concepts à Partir d'Exemples», *Proceedings of the 5th Journées Françaises de l'Apprentissage* (1990), p. 93-114.

[MAL 62] MALGRANGE, Y., «Recherche des Sous-Matrices Premières d'une Matrice à Coefficients Binaires; Applications à Certains Problème de Graphes», *Proceedings of the Deuxième Congrès de l'AFCALTI* (1962), p. 231-242.

[MIC 81] MICHALSKI, R. S., STEPP, R. E., DIDAY, E., A Recent Advance in Data Analysis: Clustering Objects into Classes Characterized by Conjunctive Concepts, in *Progress in Pattern Recognition*, L. N. Kanal, A. Rosenfeld (Eds.), North-Holland Publishing Company, p. 33-56, 1981.

[MIN 90a] MINEAU, G., GECSEI, J., GODIN, R., «Structuring Knowledge Bases using Automatic Learning», *Proceedings of the IEEE Sixth Int'l Conf. on Data Engineering* (1990), p. 274-280.

[MIN 90b] MINEAU, G., GODIN, R., GECSEI, J., «La Classification Symbolique: Une Approche Non-Subjective», *Proceedings of the 5ièmes Journées Françaises de l'Apprentissage* (1990), p. 169-190.

[MIN 92a] MINEAU, G., «Acquisition d'objets structurés destinés à la classification symbolique», *Proceedings of the Proceedings of the 1st Journées Francophones sur l'Apprentissage et l'Explicitation des Connaissances (JFAEC)* (1992).

[MIN 92b] MINEAU, G., GODIN, R., «Automatic Knowledge Structuring for Browsing Retrieval», *Proceedings of the International Conference on Information and Knowledge Management* (1992), p. 273-281.

[MIN 93] MINEAU, G., GODIN, R., MISSAOUI, R., «Induction of Generic Data Models by Conceptual Clustering», *Proceedings of the The Fifth International Conference on Software Engineering & Knowledge Engineering* (1993), p. 554-564.

[MIN 94] MINEAU, G. W., GODIN, R., «Automatic Structuring of Knowledge Bases by Conceptual Clustering», *IEEE Transactions on Knowledge and Data Engineering* , 1994, À paraître.

[NIE 90] NIELSEN, J., *Hypertext & Hypermedia*, Academic Press, 1990.

[NOR 78] NORRIS, E. M., «An Algorithm for Computing the Maximal Rectangles in a Binary Relation», *Revue Roumaine de Mathématiques Pures et Appliquées* , 23, n° 2, 1978, p. 243-250.

[OOS 92] OOSTHUIZEN, G. D., BEKKER, C., AVENANT, C., «Managing Classes in Very Large Class Repositories», *Proceedings of the Tools* (1992), p. 625-633.

[PRI 91a] PRIETO-DIAZ, R., «Implementing Faceted Classification for Software Reuse», *Communications of the ACM* , 34, n° 5, 1991a, p. 88-97.

[PRI 91b] PRIETO-DIAZ, R., ARANGO, G., Eds. *Domain Analysis and Software Systems Modeling*, IEEE Computer Society Press, 1991.

[SAL 89] SALTON, G., *Automatic Text Processing*, Addison-Wesley, 1989.

[SOW 84] SOWA, J. F., *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, 1984.

[THO 89] THOMPSON, K., LANGLEY, P., «Incremental Concept Formation with Composite Objects», *Proceedings of the Proceedings of the 6th International Workshop on Machine Learning* (1989), p. 371-374.

[WIL 82] WILLE, R., Restructuring Lattice Theory: an Approach Based on Hierarchies of Concepts, in *Ordered Sets*, I. Rival (Ed.), Reidel, p. 445-470, 1982.

[WIL 84] WILLE, R., «Line Diagrams of Hierarchical Concept Systems», *Int. Classif.* , 11, n° 2, 1984, p. 77-86.

[WIL 89] WILLE, R., Knowledge Acquisition by Methods of Formal Concept Analysis, in *Data Analysis, Learning Symbolic and Numeric Knowledge*, E. Diday (Ed.), Nova Science Pub., p. 365-380, 1989.

[WIL 92] WILLE, R., «Concept Lattices and Conceptual Knowledge Systems», *Computers Mth. Applic.* , 23, n° 6-9, 1992, p. 493-515.