# Un automate pour la génération complète ou partielle des concepts du treillis de Galois

Ganaël Jatteau, Rokia Missaoui
M. Sarifuddin

Département d'informatique et d'ingénierie
Université du Québec en Outaouais
C.P. 1250, succursale B, Gatineau
Québec, J8X 3X7
{jatg01, Rokia.Missaoui, M.Sarifuddin}@uqo.ca

**Résumé.** Cet article se situe dans le domaine de l'analyse formelle de concepts et du treillis de concepts (treillis de Galois) lequel est un cadre théorique intéressant pour le regroupement conceptuel des données et la génération des règles d'association. Puisque la prospection de données (*data mining*) est utilisée comme support à la prise de décision par des analystes rarement intéressés par la liste exhaustive (souvent très longue) des concepts et des règles, l'élaboration d'une solution approximative sera dans la plupart des cas un compromis satisfaisant et relativement moins coûteux qu'une solution exhaustive. Dans cet article, on propose une approche appelée CIGA (*Closed Itemset Generation using an Automata*) de génération partielle ou complète de concepts par la construction et le parcours d'un automate à états finis. La génération des concepts permet l'identification des "itemsets" fermés fréquents, étape cruciale pour l'extraction des règles d'association.

## 1 Introduction

L'analyse formelle de concepts (treillis de Galois) est un cadre théorique intéressant pour la prospection de données puisqu'elle permet la génération de concepts et de règles d'association. Un concept formel est un couple complet qui associe un ensemble d'objets (extension) à un ensemble d'attributs (intention) permettant ainsi de regrouper les objets qui ont des caractéristiques communes.

Dans plusieurs applications de prospection de données, la production d'un ensemble *exhaustif* de connaissances (règles d'association, concepts) peut être très coûteuse et comporter plusieurs éléments absolument peu pertinents pour un utilisateur donné. Aussi, il serait avantageux d'offrir des mécanismes de génération d'un sous-ensemble de ces connaissances qui pourraient si nécessaire inciter l'utilisateur soit à solliciter l'affichage d'autres connaissances ou à demander des détails sur les associations et les concepts issus d'un ensemble plus restreint de données.

La découverte des règles d'association se fait généralement en deux étapes : (i) la détermination de l'ensemble des "itemsets" fréquents (i.e., ceux dont le support dépasse un seuil déterminé), puis (ii) la génération des règles d'association à partir des "itemsets" fréquents obtenus à la première étape.

Des travaux de recherche relativement récents dans le domaine de la prospection de données (Bastide et al. 2003 ; Pasquier et al. 2000 ; Wang et al. 2003 ; Valtchev et al. 2002 ; Zaki 2002) ont démontré l'intérêt à produire l'ensemble des "itemsets" *fermés* fréquents lors du processus d'extraction des règles d'association. L'identification de ce nouvel ensemble d'"itemsets" assure une diminution considérable du temps d'extraction des "itemsets" fréquents et de génération des règles d'association.

Dans une démarche similaire d'exécution efficace de la première étape du processus de production des règles d'association, nous proposons une approche d'approximation du treillis de concepts ainsi qu'une nouvelle méthode de génération complète ou partielle des concepts du treillis. Notre solution est basée sur la construction et l'exploration d'un automate en vue de produire à un coût relativement faible un ensemble non nécessairement exhaustif de concepts. La sélection des concepts à garder se base non seulement sur le support des concepts (nombre d'objets possédant l'intention du concept), mais également sur les fréquences de cooccurrence d'attributs.

La section suivante rappelle les notions de base sur les treillis de concepts (Galois) et les règles d'association. La section 3 propose une technique d'approximation du treillis basée sur un automate. Finalement, des résultats expérimentaux sont présentés en section 4.

## 2 Rappels sur l'analyse formelle de concepts

| | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| 1 | x | | x | x | x | x | | |
| 2 | x | x | | | x | x | | |
| 3 | x | | x | | | | x | x |
| 4 | | | | | x | | x | |
| 5 | | x | | | | x | | |
| 6 | x | x | x | | | | | x |
| 7 | | x | | x | x | | x | |
| 8 | x | | | x | | | | |
| 9 | | x | x | x | | x | | |
| 10 | x | | x | | | | | |

TAB. 1 – Contexte avec attributs triés dans un ordre décroissant du support.

Étant donné un contexte formel $\mathcal{K} = (O, A, R)$ (voir table 1) dans lequel $R$ est une relation binaire entre un ensemble d'objets $O$ et un ensemble d'attributs $A$, l'analyse formelle de concepts (Ganter 1999) produit un treillis de concepts constitué d'un ensemble de concepts partiellement ordonnés par inclusion ensembliste. La relation binaire $R$ donne lieu à deux correspondances :

$$X' = f(X) = \{a \in A | \forall o \in X, oRa\} \; \forall X \subseteq O$$

$$Y' = g(Y) = \{o \in O | \forall a \in Y, oRa\} \; \forall Y \subseteq A$$

Le couple de fonctions $f$ et $g$ est une connexion de Galois entre $\mathcal{P}(O)$ et $\mathcal{P}(A)$.

Un concept $(X, Y)$ est défini par son extension (*extent*) $X$ et son intention (*intent*) $Y$ lesquelles sont obtenues en utilisant la fermeture de la connexion de Galois d'une relation binaire. L'extension énumère les objets du concept alors que l'intention spécifie les attributs partagés par ces objets. Cette dernière correspond à la notion d'"itemset" fermé dans l'analyse du panier du consommateur (Pasquier et al. 2000).

`Exemple :` *Dans le contexte de la table 1, les objets* 6 *et* 9 *possèdent en commun les attributs b et c. On écrit* $f(\{6,9\}) = \{6,9\}' = \{b,c\}$. *De la même façon,* $g(\{b,c\}) = \{b,c\}' = \{6,9\}$. *On obtient finalement* $\{6,9\}'' = \{b,c\}' = \{6,9\}$ *d'où {6,9} est un fermé,* $\{b,c\}$ *est un itemset fermé, et ({6,9},{b,c})*[1] *est un concept.*

De par sa structure hiérarchique, le treillis de concepts facilite la détection de régularités qui s'expriment sous la forme de règles liant deux ensembles d'attributs. La règle $r : Y_1 \Rightarrow Y_2$ indique que la présence de l'ensemble d'attributs $Y_1$ dans un objet entraîne la présence de l'ensemble $Y_2$. Elle peut être de l'un des deux types suivants : une règle d'association exacte (totale) dont la validité exige que $Y_2$ soit présent chaque fois que $Y_1$ l'est, et une règle approximative (partielle) de forme probabiliste. Deux mesures de qualité sont habituellement associées aux règles : la confiance et le support. La confiance de la règle $r : Y_1 \Rightarrow Y_2$ calcule la probabilité que $Y_2$ se produise quand $Y_1$ est présent. Le support (relatif) de la règle $r$ représente la probabilité que $Y_1$ et $Y_2$ soient simultanément présents.

# 3 Approximations du treillis en utilisant un automate

Cette section présente une famille d'automates permettant de générer en partie ou en totalité les concepts du treillis. Contrairement à une approximation "naïve"[2], l'utilisation d'un automate permet de faire une approximation "sélective" basée sur des mesures de qualité comme le support d'un concept ou la confiance d'une règle reliant deux attributs élémentaires.

## 3.1 Construction de l'automate

Il s'agit d'un automate fini et sans cycle dont chaque état représente un attribut. L'automate dispose d'un seul état initial (l'attribut vide {}) mais peut posséder plusieurs états finaux. Le langage reconnu par cet automate inclut l'ensemble des intentions qui pourront être générées. Si la manière d'exploiter un automate est générique, l'automate permettant d'identifier les concepts d'un treillis donné n'est pas unique. On préfèrera un automate minimisant le nombre de transitions entre les états afin de

[1] Pour simplifier la notation, nous utiliserons $(69, bc)$ à la place de $(\{6,9\}, \{b,c\})$.

[2] Un cas d'approximation naïve consiste à faire l'intersection des objets deux à deux pour obtenir un sous-ensemble des concepts.

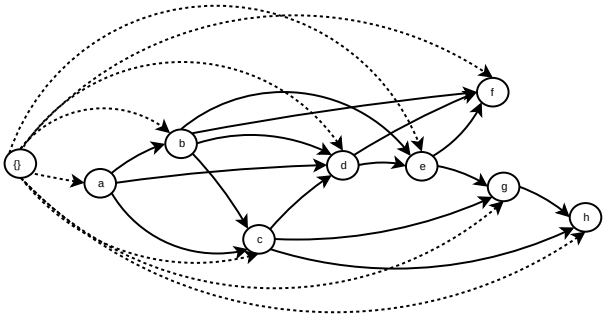


FIG. 1 – Automate complet issu directement du contexte 1 sans utilisation de la matrice des cooccurrences

rendre l'identification des concepts plus rapide. De même, il est possible d'utiliser l'automate pour générer une solution approximative en minimisant encore plus le nombre de transitions.

Selon la manière dont est généré l'automate, il est possible de faire de l'approximation sélective en se basant sur des mesures de support ou de confiance pour des règles du type $a \Rightarrow b$ $(a, b \in A)$ où $a$ et $b$ représenteront deux états reliés par une transition dans l'automate. Cela permet de limiter de manière parfois très significative le nombre de concepts générés tout en accélérant le temps d'exécution et en présentant les concepts les plus intéressants.

### 3.1.1 Matrice de cooccurrences

| | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| a | | | | | | | | |
| b | 2 | | | | | | | |
| c | 4 | 2 | | | | | | |
| d | 2 | 2 | 2 | | | | | |
| e | 2 | 2 | 1 | 2 | | | | |
| f | 2 | 3 | 2 | 2 | 2 | | | |
| g | 1 | 1 | 1 | 1 | 2 | 0 | | |
| h | 2 | 1 | 2 | 0 | 0 | 0 | 1 | |
| card. | 6 | 5 | 5 | 4 | 4 | 4 | 3 | 2 |

TAB. 2 – Matrice des cooccurrences d'attributs

La construction de l'automate se base sur l'élaboration d'une matrice de cooccurrences d'attributs dont le coût de génération se fait en temps polynomial. Le calcul

du support ou de la confiance pour chaque règle se fait à l'aide de cette matrice (voir table 2).

On part d'un contexte trié pour construire cette matrice, ce qui permet de traiter en premier lieu les attributs qui ont le plus grand support et donc plus de chance d'avoir des liens avec d'autres attributs. La cellule $(i, j)$ de la matrice indique le nombre de fois que l'attribut $a_i$ se retrouve avec l'attribut $a_j$ dans la description des objets. La matrice peut être utilisée directement pour identifier le support ou pour estimer la confiance d'une règle entre deux attributs individuels. Par exemple, sachant que $f$ intervient trois fois avec $b$ et que $b$ est présent cinq fois dans le contexte (voir table 2), la confiance de $b \Rightarrow f$ est de 3/5 (=60%). La table 3 répertorie chacune des confiances en pourcentage. L'arbre TRIE ci-après (Cf. figure 2) est une autre façon de représenter le contexte et de mettre en relief les cooccurrences d'attributs.

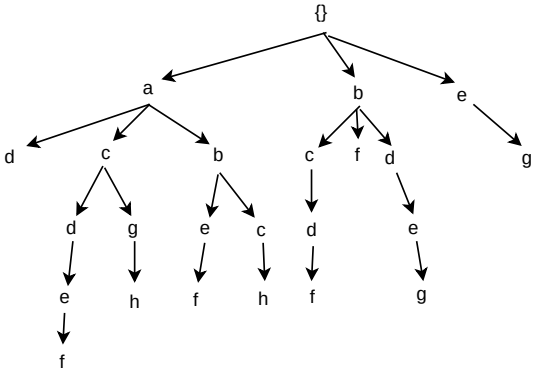


FIG. 2 – Arbre TRIE issu du contexte de la table 2

`Exemple :` *L'arbre TRIE (fig. 2) permet de voir que f existe trois fois en présence de b dans le contexte. Cette information est traduite dans la matrice de cooccurrences (table 2) par la cellule de coordonnées (f,b).*

Afin d'alléger l'automate et réduire en conséquence les états visités, les cellules de la matrice (table 2) ayant des valeurs au plus égales à 1 peuvent être ignorées lors de la construction de l'automate . Elles correspondent en fait à des binômes d'attributs apparaissant une seule fois dans le contexte et donc ne présentant pas un intérêt particulier lors de la recherche d'intersections entre les intentions. En effet, une intention est soit issue d'un concept objet (c-à-d le concept le plus spécifique contenant un objet donné) ou d'une intersection entre un ou plusieurs concepts objets. Dans le premier cas, la découverte des concepts objets n'est pas compliquée puisque ces concepts apparaissent directement dans le contexte. Dans le deuxième cas, pour qu'un tel type de concept existe, il faut qu'il apparaisse au moins dans deux branches de l'arbre TRIE (figure 2)

des intentions. Sachant que toutes les intentions sont présentes dans une intersection d'une ou plusieurs branches de l'arbre TRIE, on ne retiendra que les règles d'association présentes au moins deux fois dans la matrice des supports (table 2) laissant ainsi de côté les concepts à un seul objet. Ces règles sont matérialisées par des valeurs de cellules supérieures ou égales à 2. Par exemple, la règle $c \Rightarrow e$ n'existe qu'une seule fois dans la table ce qui signifie que les attributs $c$ et $e$ ne peuvent apparaître ensemble que dans un concept objet dont le support absolu est 1. Il n'est donc pas nécessaire de mentionner cette règle lors de la génération de l'automate.

### 3.1.2 Construction de l'automate et paramètres d'approximation

L'automate de la figure 3 a été généré à partir du contexte de la table 3 en se limitant aux cellules ayant une confiance minimale de 35%. On dira que l'automate est complet s'il contient dans son langage l'ensemble des intentions issues d'un contexte. Il existe de nombreuses façons de générer cet automate.

| | {} | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|---|
| a | 60 | | | | | | | | |
| b | 50 | 33 | | | | | | | |
| c | 50 | 66 | 40 | | | | | | |
| d | 40 | 33 | 40 | 40 | | | | | |
| e | 40 | 33 | 40 | 20 | 50 | | | | |
| f | 40 | 33 | 60 | 40 | 50 | 50 | | | |
| g | 30 | 16 | 20 | 20 | 25 | 50 | 0 | | |
| h | 20 | 33 | 20 | 40 | 0 | 0 | 0 | 33 | |

TAB. 3 – Table des confiances en %

La manière la plus simple de procéder consiste à prendre directement les concepts objets issus du contexte (automate de la figure figure 1). Par exemple, le concept objet $(\{6\}, \{abch\})$ issu du contexte de la table 1 se traduit par la création du lien $\{\} \Rightarrow a \Rightarrow b \Rightarrow c \Rightarrow h$ où $\{\}$ représente nécessairement l'état initial. Cette solution offre toutefois peu d'intérêt en raison du nombre important de transitions générées.

Le rôle de la matrice de cooccurrences est justement d'éliminer les transitions les moins significatives. Sachant qu'une transition peut toujours être assimilée à une règle du type $a \Rightarrow b, \forall a, b \in A$, on peut construire l'automate à partir des règles les plus significatives (selon le support ou la confiance). L'automate obtenu va donc permettre de générer une approximation du treillis par la production d'un ensemble de concepts les plus utiles et pertinents. Ce genre d'approximation est similaire à la notion d'iceberg (Stumme et al., 2002).

`Exemple :` *En construisant l'automate (fig. 3), on décide d'utiliser les confiances supérieures à 35%. La transition $d \Rightarrow g$ n'apparait pas dans l'automate puisqu'elle possède une confiance de 25%. En revanche, les transitions $d \Rightarrow e \Rightarrow g$ apparaitront ; le seul concept contenant $d$ et $g$ étant* $(7, bdeg)$.

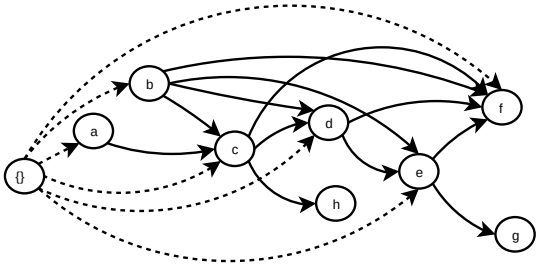


FIG. 3 – Automate généré à partir de la table 3 avec une confiance minimale de 30%

Les méthodes d'approximation à partir de l'automate peuvent être nombreuses. On propose dans cet article de se baser sur le support ou la confiance des règles. D'autres méthodes sont envisageables telles que la limitation au nombre $n$ de transitions sortantes d'un état. On peut ainsi limiter le parcours de l'automate à un arbre n-aire dans lequel on ne sélectionne que les transitions de plus fortes confiances.

## 3.2 Parcours de l'automate avec CIGA

L'exploitation de l'automate se fait de manière récursive. On part de l'état initial pour accéder aux différents états de l'automate. Cette méthode revient à étaler l'automate sous forme d'un arbre pour explorer l'ensemble de ses branches. Un chemin de l'origine vers un état donné représente un itemset qui peut être fermé (notion d'intention de concept) ou non. L'algorithme 2 décrit son fonctionnement.

Pour chaque état de l'automate, on calcule la correspondance $g(Y) = Y'$ où $Y$ est un ensemble d'attributs représentant le chemin parcouru de l'état initial à l'état courant. En se basant sur le fait que $\forall Y \in \mathcal{P}(A)$, $\forall a \in A$, $g(Y \cup a) = g(Y) \cap g(a)$, le calcul de $g(Y \cup a)$ à l'état représenté par $a$ ne nécessite pas de calcul de la correspondance $g$ au complet et se base donc sur la valeur déjà calculée de $g(Y)$ et celle de $g(a)$. Le passage d'un état à un autre revient à faire seulement une intersection entre deux ensembles d'objets, ce qui rend la procédure efficace.

La progression d'un état vers un autre est stoppée dans les deux cas suivants :

1. On arrive à l'état terminal.
2. $X = g(Y) \leq sup\_min$ ($sup\_min$ étant le support minimal souhaité).

Lorsque l'automate est complet (l'ensemble des intentions est compris dans son langage), on peut déterminer la fermeture d'une intention par la propriété suivante :

$$\forall Y \in \mathcal{P}(A), \forall a \in A - Y, Y = Y'' \iff g(Y \cup a) \subset g(Y)$$

Cela signifie que si l'ajout d'un attribut quelconque à une intention $Y$ modifie la valeur de sa fonction de correspondance $g$, alors l'intention $Y$ est fermée. On peut utiliser cette propriété simplement en testant la cardinalité $card(X = g(Y)$ pour chaque état

successeur. Si cette taille est toujours différente par rapport à l'intention courante, alors on conclut que cette dernière est fermée (évitant ainsi de vérifier la fermeture).

Lorsque l'automate n'est pas complet, la propriété précédente devient une condition nécessaire mais non suffisante. Lorsqu'elle est vérifiée, le concept actuel devient un candidat intéressant et le calcul de sa fermeture est nécessaire (ligne 7 dans l'algorithme 1).

**Algorithme 1** CIGA2

`Procédure` CIGA2
**Entrée :** $a$, $X$, $Y$
**Sortie :** $X$ (extension courante)
1: $X \leftarrow g(a) \cap X$
2: candidat = vrai
3: **si** $supp\_min < \frac{X.taille}{|O|}$ **alors**
4: **pour** chaque état $fils$ de $a$ **faire**
5: $X_{fils} \leftarrow$ CIGA2($fils$, $X$, $Y \cup fils$)
6: **si** ($X_{fils}.taille = X.taille$) **alors** candidat = faux
7: **si** (candidat) **et** $f(X) = Y$ **alors** $L \leftarrow L \cup (X, Y)$
8: **retourner** $X$

**Algorithme 2** CIGA

**Entrée :** Automate $A$, $supp\_min$
**Sortie :** Liste des concepts $L$
1: $L \leftarrow \emptyset$
2: **pour chaque** état $a$ de $A$ **faire**
3: CIGA2(a,g(a),a)
4: **retourner** $L$

`Signification des variables dans les procédures CIGA et CIGA2 :`
– *a : état de l'automate actuellement parcouru*
– *X : extension associée à l'état parcouru*
– *Y : intention associée à l'état parcouru*
– *|0| : nombre d'objets dans le contexte*
– *supp_min : support minimal.*

`Exemple :` *Les concepts générés à partir de l'automate de la figure 3 sont : (36,ach), (136 10, ac), (69,bc), (79,bd), (27,be), (259,bf), (19,cdf), (17,de), (47,eg)*[3]*. Les concepts écartés sont (12,aef), (26,ab), (18,ad). Cet exemple a requis 44 appels récursifs de la procédure CIGA2.*

[3]La liste exclut les concepts ayant un seul objet ou un seul attribut. Ces concepts apparaissent déjà dans le contexte.

# 4 Expérimentations

Les expérimentations ont été menées sur la plate-forme Galicia [4] dans un environnement Java (Pentium IV, 3Ghz, 1Go de RAM) en utilisant la base de données *Mushroom*[5]. Cette dernière est un ensemble relativement dense de données sur les particularités d'espèces de champignons et comporte 8124 objets, 120 attributs, et un nombre moyen de 23 attributs par objet.

L'algorithme CIGA est difficilement comparable avec d'autres algorithmes existants. Toutefois, lorsque l'automate est construit avec une confiance égale à zéro, CIGA produit l'ensemble des fermés fréquents pour un support minimal donné.

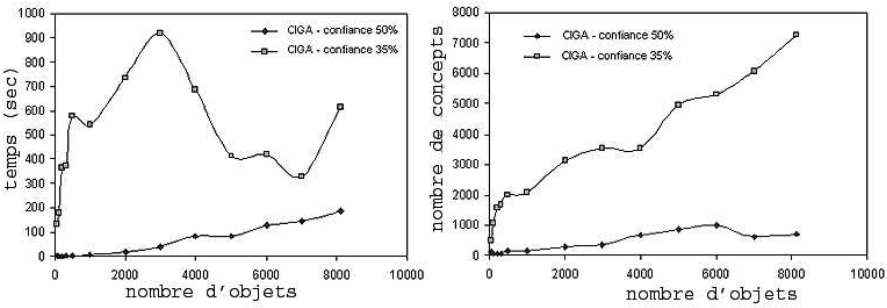


FIG. 4 – CIGA testé sur la base de données *Mushroom* avec support minimal de 1%

Les résultats de la figure 4 indiquent les performances de CIGA selon le nombre d'objets pour un support et une confiance donnés. On remarque tout d'abord que le temps d'éxécution n'évolue pas toujours de façon croissante avec le nombre d'objets. Sachant que l'on cherche à faire une approximation à partir de la confiance (valeur relative), il est possible que des informations dans le contexte deviennent pertinentes lorsque le nombre d'objets diminue, ce qui a pour conséquence une augmentation du temps d'exécution. Quant au nombre de concepts calculés, il dépend énormément de la confiance définie par l'analyste.

Comme on pouvait s'y attendre, l'évolution de l'effectif des concepts selon le nombre d'objets est décroissante en fonction de la confiance. Pour une confiance importante (au delà de 50%), le nombre de concepts reste toujours faible quelque soit le nombre d'objets. Définir une confiance élevée offre deux avantages : (i) obtention d'un sous-ensemble réduit des concepts et (ii) amélioration du temps d'éxécution de l'algorithme. La figure 5 indique l'influence de la confiance sur la base *Mushroom* par rapport au temps d'exécution et au nombre de concepts calculés. Il est possible d'obtenir un compromis intéressant vis à vis du nombre de concepts générés en choisissant habilement la confiance.

[4] *Galois Lattice Interactive Constructor* - http ://www.iro.umontreal.ca/~galicia/.
[5] *Frequent Itemset Mining Dataset Repository* - http ://fimi.cs.helsinki.fi/data/.

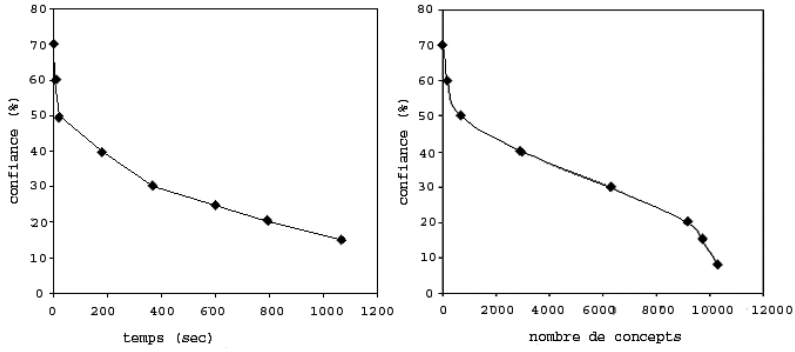


FIG. 5 – Influence de la confiance dans CIGA (*Mushroom* avec support minimal de 5% et temps de prétraitement non inclus)

Ces résultats préliminaires laissent présager un potentiel intéressant de la procédure CIGA dans l'extraction sélective des concepts et des règles d'association.

# 5 Conclusion

Nous avons présenté une méthode de calcul d'un ensemble (partiel ou total) de concepts du treillis de Galois basée sur l'utilisation d'un automate. Des expérimentations préliminaires montrent l'efficacité de la méthode à gérer un volume important de données en présence de contextes (ou bases de transactions) creux ou denses. Comme l'intention d'un concept n'est rien d'autre qu'un "itemset" fermé fréquent, la méthode peut être appliquée non seulement à la construction d'un treillis de Galois (ou d'un iceberg de treillis lorsqu'un seuil de support est retenu), mais également à la génération des "itemsets" fermés fréquents, étape cruciale du processus d'extraction des règles d'association. La méthode pourra également servir pour faire de la prospection de données "à la carte" (à la demande de l'utilisateur) afin de tester par exemple l'existence d'un concept ou d'un "itemsets" fermé fréquent, ou la présence d'une règles d'association vérifiant une condition donnée.

Nos travaux futurs visent la complétion de notre méthode pour qu'elle puisse également produire l'ordre partiel entre les concepts et être comparée à d'autres procédures de construction du treillis de Galois. Nous chercherons également à adapter et enrichir nos travaux antérieurs sur l'extraction de générateurs de concepts et des règles d'association (Valtchev et al. 2002, Valtchev et al. 2003) en vue de rendre notre solution complète au niveau de la prospection de données. Des expérimentations plus intensives seront conduites ultérieurement sur diverses bases de données.

## Références

Bastide (2002), Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L., "PASCAL : un algorithme d'extraction de motifs fréquents", Technique et Science Informatiques, vol. 21, n 1, 2002, p. 65-95.

Birkhoff (1967), Birkhoff, G., Lattice theory. In Colloquium Publications, volume 25. American Mathematical Society, 1967, Third edition.

Burdick (2001), D. Burdick, M. Calimlim, et J. Gehrke. MAFIA : a maximal frequent itemset algorithm for transactional databases. In Proceedings of the 17th IEEE ICDE Conference (ICDE01), pages 443-452, Heidelberg, Germany, April.

Ganter (1984), Ganter, B. Two basic algorithms in concept analysis (preprint). Technical Report 831, Technische Hochschule, Darmstadt.

Ganter (1999), B. Ganter, R. Wille : Formal Concept Analysis : Mathematical Foundations. Springer, Berlin-Heidelberg 1999

Godin (1994), R. Godin et R. Missaoui, An incremental concept formation approach for learning from databases. Theoretical Computer Science, volume 133, pages 387-419.

Godin (1995), Godin, R., Missaoui, R. et Alaoui, H. Incremental concept formation algorithms based on galois lattices. Computational Intelligence, 11(2) :246-267.

Lakhal (1999), Lotfi Lakhal, Nicolas Pasquier, Yves Bastide, Rafik Taouil. Efficient mining of association rules using closed itemset lattices. Volume 24 , Pages : 25 - 46, ISSN :0306-4379

Pasquier (2000), Pasquier, N., Data mining : algorithmes d'extraction et de réduction des règles d'association dans les bases de données. Thèse de doctorat, Université de Clermont-Ferrand II, 2000.

Stumme (2002), Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L., Computing Iceberg Concept Lattices with Titanic. Data and Knowledge Engineering, 42(2), 189-222, 2002.

Valiant (1979), Valiant, L. G. The complexity of enumeration and reliability problems, SIAM Journal on Computing, 8, pp. 410-421.

Valtchev (2001), P. Valtchev et R. Missaoui, Building concept (Galois) lattices from parts : generalizing the incremental methods. In H. Delugach et G.Stumme, editors, Proceedings, ICCS-01, Lecture Notes in Computer Science, pages 290-303, Stanford (CA), USA. Springer-Verlag.

Valtchev (2002), P. Valtchev, R. Missaoui, R. Godin, et M. Meridji. Generating Frequent Itemsets Incrementally : Two Novel Approaches Based On Galois Lattice Theory. Journal of Experimental and Theoretical Artificial Intelligence, 14(2-3), p. 115-142.

Valtchev (2004), Petko Valtchev, Rokia Missaoui, Robert Godin : Formal Concept Analysis for Knowledge Discovery and Data Mining : The New Challenges. Proc. of International Conf. on Formal Concept Analysis (ICFCA) 2004 : 352-371.

## Remerciements

La réalisation de cette recherche a été rendue possible grâce à la contribution financière du CRSNG (Conseil de recherches en sciences naturelles et en génie du Canada) et du consortium CoRIMedia.

## Summary

This paper deals with formal concept analysis and concept (Galois) lattices which represent a theoretical and interesting framework for conceptual clustering and association rule mining. Since data mining is used as a mechanism for decision support by analysts who are rarely interested by the exhaustive (and frequently large) set of concepts and association rules, an approximate solution is a more satisfactory and cost-effective compromise than an exhaustive one. In this paper, we propose an approach called CIGA (Closed Itemset Generation Using an Automata) towards the partial or complete generation of concepts based on the construction and exploration of a fine-state machine. Such concept extraction allows the identification of frequent closed "itemsets", an important step towards association rule mining.