MATHÉMATIQUES ET SCIENCES HUMAINES

A. GUÉNOCHE

Construction du treillis de Galois d'une relation binaire

Mathématiques et sciences humaines, tome 109 (1990), p. 41-53

<http://www.numdam.org/item?id=MSH_1990__109__41_0>

Math. Inf. Sci. hum. (28e année, n°109, *1990*, p.41-53)

CONSTRUCTION DU TREILLIS DE GALOIS D'UNE RELATION BINAIRE

A. GUÉNOCHE[1]

RÉSUMÉ - *Cet article constitue une présentation unifiée des principales méthodes de construction du treillis de Galois d'une correspondance. Nous rappelons d'abord sa définition, puis nous décrivons quatre algorithmes de construction des éléments du treillis qui sont les rectangles maximaux de la relation binaire. Ces algorithmes ne sont pas originaux. Les descriptions précises de algorithmes, le plus souvent absentes des publications originales, permettent une programmation simple, dans un langage procédural à l'aide d'une structure de données commune.*

ABSTRACT - Building the Galois Lattice of a binary relation
This text is a survey of different methods used to build the Galois Lattice of a binary relation between two finite sets. We first recall its definition and common notations. Then we present four algorithms to construct the elements of the lattice that are maximal rectangles in the binary relation. These algorithms have already been published during these last twenty years. Precise descriptions, often missing in the original publications, are given and permit a simple programming job in any procedural language.

L'intérêt pour le treillis de Galois d'une correspondance binaire entre deux ensembles s'est trouvé réactivé, à cause de la position centrale qu'il occupe dans certains problèmes relevant à la fois de la Théorie de l'apprentissage à partir d'exemples et de l'Analyse combinatoire de données. Depuis quelques années, on s'intéresse à cette structure, équivalente à celle de la correspondance, dans le cadre de la définition de concepts [Wille 1982, 1989] et de la recherche d'implications [Duquenne et Guigues 86]. Ce texte ne contient pas de résultats mathématiques nouveaux; il s'agit d'une contribution informatique dans laquelle on présente les idées qui conduisent aux algorithmes de construction du treillis de Galois. Nous rappelons d'abord sa définition, puis nous décrivons quatre algorithmes de construction des éléments du treillis qui sont les rectangles maximaux de la relation binaire. Ces algorithmes ne sont pas originaux. Cet article constitue une présentation unifiée des principales méthodes de construction du treillis. Les descriptions précises des algorithmes, absentes des publications originales, à l'exception de celle de Bordat [1986], permettent une programmation simple, dans un langage procédural, à l'aide d'une structure de données commune.

1. INTRODUCTION

Nous ne prétendons pas dans cette introduction rappeler au lecteur les éléments de base de la théorie mathématique des treillis, pas plus que nous ne démontrerons les propriétés élémentaires qui amènent à la définition du treillis de Galois associé à une correspondance entre deux

[1] G.R.T.C. - C.N.R.S. 31 Ch. J. Aiguier 13402 Marseille Cedex 9

ensembles finis. Le lecteur intéressé consultera les ouvrages de référence [Birkhoff 1967, Barbut & Monjardet 1970]. Nous présentons les seules notions nécessaires à la définition des algorithmes et introduisons les notations les plus usuelles dans ce domaine.

Soit E = {1, 2, ... , n} et F = {a, b, c, ...} deux ensembles de cardinaux respectifs n et m et R une relation binaire sur ExF. On notera xRy le fait qu'un élément x de E est lié à un élément y de F. Définissons :

$$f : \mathcal{P}(E) \ni X \mapsto f(X) = \{y \in F \mid \forall\, x \in X, xRy\} \subset F$$
$$g : \mathcal{P}(F) \ni Y \mapsto g(Y) = \{x \in E \mid \forall\, y \in Y, xRy\} \subset E.$$

De façon équivalente f(X) est l'intersection pour x appartenant à X de f(x) = {y ∈ F | xRy} et g(Y) l'intersection pour y appartenant à Y de g(y) = {x ∈ E | xRy}. A l'évidence f (resp. g) est une application monotone décroissante.

$$X \subset X' \Rightarrow f(X') \subset f(X).$$

On note h = g[f] et h' = f[g] les applications composées de f et g. Ce sont des applications :

- monotones croissantes $X \subset X' \Rightarrow f(X') \subset f(X) \Rightarrow h(X) \subset h(X')$,
- extensives $X \subset h(X)$,
- idempotentes $h[h(X)] = h(X)$.

Par définition le couple {f, g} forme une *correspondance de Galois* et h et h' sont des *fermetures*. Un sous-ensemble X de E (resp. Y de F) est dit *fermé* si et seulement si X = h(X) (resp. Y = h'(Y)).

Soit $\mathfrak{L}_E$ (resp. $\mathfrak{L}_F$) l'ensemble des fermés de E (resp. F) partiellement ordonné par inclusion. Pour les opérations d'union (infimum) et d'intersection (supremum) ($\mathfrak{L}_E$, $\subset$) a une structure de treillis, ainsi que ($\mathfrak{L}_F$, $\subset$). Ces deux treillis sont isomorphes. Notons $\mathfrak{L} = \mathfrak{L}_E \times \mathfrak{L}_F$ les couples d'éléments en correspondance dans cet isomorphisme ; ils sont de la forme X x f(X) ou g(Y) x Y.

On munit $\mathfrak{L}$ de la relation d'ordre (notée $\langle$) définie par

$$X \times Y \langle X' \times Y' \text{ si et seulement si } X' \subset X \text{ et } Y \subset Y'.$$

Le treillis ($\mathfrak{L}$, $\langle$) est appelé *treillis de Galois* de la relation R sur E x F.

PROPOSITION 1. Les éléments de $\mathfrak{L}$ sont des rectangles maximaux de la relation R.

DEMONSTRATION. La correspondance R peut se représenter sous forme d'un tableau binaire. Le sous-tableau correspondant à X x f(X), élément du treillis ne contient que des 1 puisque pour tout x ∈ X et tout y ∈ f(X), on a xRy. Supposons qu'il existe x' ∈ E/X tel que f(X) ⊂ f(x'). Alors x' ∈ g[f(X)] et f(X) n'est pas fermé. Le sous-tableau correspondant à un élément du treillis $\mathfrak{L}$ est donc maximal au sens suivant : si on lui ajoute une ligne ou une colonne, le sous-tableau étendu contiendra au moins un 0.

Ces rectangles sont également appelés *sous-matrice complètes premières* [Kaufmann et Pichat 1977]. Dans une terminologie différente, issue de la théorie des graphes et que nous ne détaillerons pas ici, la relation R définit les arêtes d'un graphe biparti sur E∪F, f(x) (resp. g(y))

est l'ensemble des sommets adjacents à x (resp. y). Les éléments de $\mathfrak{L}$ correspondent aux sous-graphes bipartis complets maximaux (les cliques maximales) du graphe (E∪F, R).

EXEMPLE 1
Soit E = {1,2,3,4,5,6}, F = {a,b,c,d,e} et R donnée par le tableau binaire ci-dessous.

	a	b	c	d	e
1	1	1	0	0	0
2	1	0	1	0	0
3	0	1	1	1	0
4	0	0	0	1	1
5	0	0	0	1	0
6	1	1	1	1	1

On a f(1) = {a,b} et g({a,b}) = {1,6}, d'où h(1) = {1,6} et {1} n'est pas fermé. Par contre f({1,6} = {a,b} d'où h ({1,6}) = {1,6} et {1,6} est fermé : {1,6} x {a,b} est un élément du treillis de Galois. En anticipant quelque peu, nous présentons dans la figure 1 le *diagramme de Hasse* du treillis, c'est à dire le graphe dont l'ensemble des sommets est l'ensemble des éléments du treillis et les arcs correspondent à la relation de couverture de la relation d'ordre.

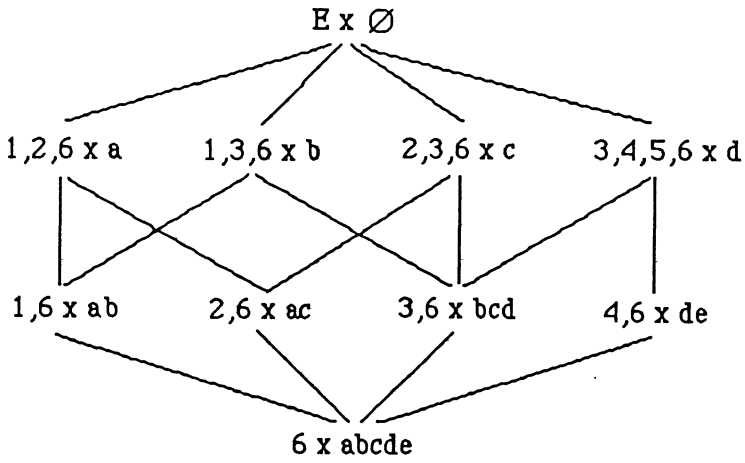

Figure 1

Nous avons tracé dans la figure 2 le graphe biparti de la relation R ainsi que l'un de ses sous-graphes complets bipartis maximaux correspondant à l'élément {3,6} x {b,c,d} de $\mathfrak{L}$.

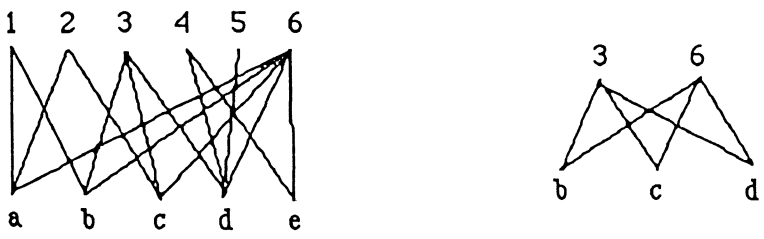

Figure 2.

2. ALGORITHMES DE CONSTRUCTION DU TREILLIS DE GALOIS

Dans cette partie nous présentons successivement quatre algorithmes publiés qui permettent de construire le treillis de Galois d'une correspondance. Nous ne reprenons pas l'algorithme "naïf" qui consiste à énumérer $\mathcal{P}(F)$, à calculer la fermeture de chaque partie pour vérifier s'il s'agit d'un fermé, non plus que des algorithmes généraux d'énumération des cliques maximales d'un graphe quelconque. Il nous a paru intéressant de reprendre et d'unifier les différentes idées développées pour cette construction dans la mesure où on les trouve dans la littérature de façon très éparse, aussi bien dans le temps et dans les revues et/ou actes de colloque que dans les terminologies et notations utilisées. D'autres publications mentionnées dans la bibliographie ne sont pas détaillées ici, soit qu'elles ne présentent pas de principes algorithmiques réellement différents de ceux exposés ci-dessous, soit que les méthodes proposées le sont de façon tellement vague qu'on ne peut parler vraiment d'algorithme. Les trois premiers algorithmes présentés ici ne s'intéressent qu'à l'énumération des éléments du treillis (les rectangles maximaux), laissant de côté le calcul de la *partie essentielle* de la relation d'ordre entre éléments, donc la construction effective du *graphe de Hasse* du treillis (encore appelé *graphe de couverture*). Le quatrième construit les différents éléments en utilisant cette relation de couverture ; il énumère donc les arêtes et conduit directement à l'algorithmique sur les graphes (construction de chemins, ou tracé automatique).

2.1 *Méthode de Chein / Malgrange*

La méthode ici attribuée à M. Chein est une amélioration de l'algorithme de Y. Malgrange, du moins tel qu'il est décrit dans l'ouvrage de Kaufmann et Pichat. Cette méthode a été utilisée (et souvent non publiée) par de nombreux auteurs (potentiels) ; c'est celle que l'on utilise pour une construction "à la main". La publication de 1962 est à ma connaissance la première qui propose une méthode spécifique ; celle publiée par G. Fay est similaire. L'algorithme est fondé sur la propriété suivante dont la démonstration est évidente.

PROPOSITION 2. Soit $X_1 \times Y_1$ et $X_2 \times Y_2$ deux éléments de $\mathfrak{L}$. Le rectangle $(X_1 \cup X_2) \times (Y_1 \cap Y_2)$ est un élément de $\mathfrak{L}$ si et seulement si il est maximal.

L'algorithme qui en découle est itératif. A chaque étape k, on part d'un ensemble L^k de rectangles et on construit L^{k+1}. Un élément de L^{k+1} est obtenu par combinaison de deux éléments de L^k, comme l'union des deux parties de E x l'intersection des deux parties de F si celle-ci n'est pas vide. Les éléments de L^k inclus dans au moins un élément de L^{k+1} ne sont pas maximaux et ils sont supprimés. Si L^{k+1} a plus d'un élément on passe à l'étape k+1, sinon l'algorithme s'arrête. L'ensemble des éléments non supprimés des différents L^k est celui des rectangles maximaux de la correspondance. Initialement L^1 est l'ensemble des rectangles à une ligne correspondant à chaque élément x_i de E, les colonnes étant les éléments de F qui lui sont liés ($\{f(x_i)\}$).

Algorithme de Chein

$L^1 := x_i \times f(x_i)_{i=1,..n}$
$k := 1$
Tant que $|L^k| > 1$ Faire
  $L^{k+1} := \varnothing$
  Pour tout $i < j$ indices d'éléments de L^k non marqués Faire
    $Y_{ij} := Y_i \cap Y_j$

Si $Y_{ij} \neq \varnothing$ Faire

Si $Y_{ij} \in L^{k+1}$ Alors $X_{ij} := X_{ij} \cup X_j$

Sinon $L^{k+1} := L^{k+1} + X_i \cup X_j \times Y_{ij}$

Si $Y_{ij} = Y_i$ Alors marquer $X_i \times Y_i$ de L^k

Si $Y_{ij} = Y_j$ Alors marquer $X_j \times Y_j$ de L^k

Fin de condition si

Fin de la boucle i < j

Fin de la boucle Tant que

$\mathfrak{L}$ est l'ensemble des éléments non marqués.

EXEMPLE 2

Dans tous les exemples, nous reprenons les données de l'exemple 1. Les différents étapes du déroulement de l'algorithme figurent ci-dessous. On initialise L^1 avec les rectangles $x_i \times f(x_i)$.

1	x	a, b	*	|	
2	x	a, c	*	|	
3	x	b, c, d	*	|	L^1
4	x	d, e	*	|	
5	x	d	*	|	
6	x	a, b, c, d, e		|	
1,2	x	a	*	|	
1,3	x	b	*	|	
1,6	x	a, b		|	
2,3	x	c	*	|	
2,6	x	a, c		|	L^2
3,4	x	d --> 3, 4, 5 x d	*	|	
3,6	x	b,c,d		|	
4,6	x	d, e		|	
1, 2,6	x	a		|	
1, 3,6	x	b		|	L^3
2, 3,6	x	c		|	
3, 4, 5, 6	x	d		|	

Les deux éléments de L^1, 1 x a,b et 6 x a,b,c,d,e donnent 1,6 x a,b ; comme l'intersection donne même partie de F que le premier, 1 x a,b est marqué (par le signe *). La composition de 3 x b,c,d avec 4 x d,e donne 3,4 x d. Puis celle de 3 x b,c,d avec 5 x d donne la même partie de F. Le rectangle 3,4 x d est prolongé en 3,4,5 x d.

COMPLEXITÉ

Notons l_k le cardinal de L^k. L'implémentation de cet algorithme utilise deux listes consécutives L^k et L^{k+1}. Le nombre d'opérations à chaque étape est proportionnel à l_k^2, puisqu'on conjugue les éléments de L^k deux à deux. Chaque combinaison (union et intersection) demande $|E|+|F| = n+m$ opérations. Vérifier que l'intersection des parties de F n'a pas déjà été obtenue dans L^{k+1} se fait en l_{k+1} comparaisons des valeurs d'une fonction caractéristique. Chaque itération est en $O(l_k^2.l_{k+1}.(n+m))$. Il s'agit d'une borne supérieure puisque le jeu des marquages entraîne un nombre nettement plus petit de comparaisons.

2.2 *Méthode de Ganter* (1984)

Cet algorithme construit tous les fermés pour une relation de fermeture quelconque; il a été développé dans le cadre de la "concept analysis". Présentons tout d'abord les notations nécessaires :

Soit $A = (a_1, a_2, .. a_m)$ et $B = (b_1, b_2, .. b_m)$ deux vecteurs caractéristiques de $\mathcal{P}(F)$ et on note $<$ l'ordre lexicographique sur ces vecteurs booléens; par exemple $(0,0,0,0) < (0,0,0,1) < (0,0,1,0) < (0,0,1,1) < (0,1,0,0)$ ..etc. Soit $A < B$ et i l'indice tel que pour tout $j < i$ on ait $a_j = b_j$ et $a_i < b_i$. (on a $a_i = 0$ et $b_i = 1$). On note $A <_i B$. Maintenant si dans $A = (a_1, a_2, .. a_m)$ on a $a_j = 0$, on note $\underline{A}_j = (a_1, .. a_{j-1}, 1, 0, .. 0)$ et on définit $A \oplus j = h'(\underline{A}_j)$. B. Ganter en déduit trois lemmes et une proposition que nous ne redémontrerons pas :

- $A < A \oplus i$
- $A <_i B$ et B fermé $\Rightarrow A \oplus i \leq B$
- $A <_i B$ et B fermé $\Rightarrow A <_i A \oplus i$.

PROPOSITION 3. Le plus petit fermé placé après A pour l'ordre lexicographique est $A \oplus i$ où i est le plus grand indice tel que $A <_i A \oplus i$.

L'algorithme de B. Ganter est fondé sur cette proposition. En partant d'un fermé A quelconque, le dernier obtenu, il détermine le plus petit fermé (pour l'ordre lexicographique) suivant A. Pour ce faire, il change une composante d'indice i le plus grand possible et telle que $a_i = 0$. Pour le vecteur $\underline{A}_i$ ainsi obtenu, il calcule sa fermeture et soit j tel que $A <_j h'(\underline{A}_i)$. Si $i<j$ alors $h'(\underline{A}_i)$ est le plus petit fermé (pour l'ordre lexicographique) suivant A, sinon on essaye une nouvelle valeur de i plus petite. Initialement le plus petit fermé de F est $\varnothing$; on partira donc de $A = (0, 0, .. 0)$.

Algorithme de Ganter

```
A := (0, 0, .. 0)
Tant que A ≠ (1, 1, .. 1) Faire
    i := m
1/  Si a_i = 1 Faire i := i - 1 ; Aller en 1/
    Sinon Faire
          Pour j := i+1 à m Faire a_j = 0
          A' := h'(A)
          Pour j := 1 à i-1 Faire
                Si a_j < a'_j  Alors i := i - 1 ; Aller en 1/
          Fin de boucle j
          A := A'
Fin de boucle Tant que
```

EXEMPLE 3

Enumérons les fermetures calculées par l'algorithme de Ganter sur les données de l'exemple 1 :

```
h'(0,0,0,0,1)  =  (0,0,0,1,1)
h'(0,0,0,1,0)  =  (0,0,0,1,0)  d'où {d} est fermé
h'(0,0,0,1,1)  =  (0,0,0,1,1) d'où {d,e} est fermé
h'(0,0,1,0,0)  =  (0,0,1,0,0) d'où {c} est fermé
h'(0,0,1,0,1)  =  (1,1,1,1,1)
h'(0,0,1,1,0)  =  (0,1,1,1,0)
h'(0,1,0,0,0)  =  (0,1,0,0,0) d'où {b} est fermé
h'(0,1,0,0,1)  =  (1,1,1,1,1)
h'(0,1,0,1,0)  =  (0,1,1,1,0)
```

h'(0,1,1,0,0) = (0,1,1,1,0) d'où {b,c,d} est fermé
h'(0,1,1,1,1) = (1,1,1,1,1)
h'(1,0,0,0,0) = (1,0,0,0,0) d'où {a}est fermé
h'(1,0,0,0,1) = (1,1,1,1,1)
h'(1,0,0,1,0) = (1,1,1,1,1)
h'(1,0,1,0,0) = (1,0,1,0,0) d'où {a,c} est fermé
h'(1,0,1,0,1) = (1,1,1,1,1)
h'(1,0,1,1,0) = (1,1,1,1,1)
h'(1,1,0,0,0) = (1,1,0,0,0) d'où {a,b} est fermé
h'(1,1,0,0,1) = (1,1,1,1,1)
h'(1,1,0,1,0) = (1,1,1,1,1)
h'(1,1,1,0,0) = (1,1,1,1,1) d'où {a,b,c,d,e} est fermé.

On a donc calculé 21 fermetures, au lieu des 31 que l'on aurait calculées pour l'algorithme "naïf" qui consiste à construire les fermetures des 2^m-1 parties de F (toutes sauf l'ensemble vide).

REMARQUES
- Cet algorithme ne calcule que les fermés de F. Pour obtenir les éléments du treillis de Galois, il faudrait, pour chaque fermé A', calculer g(A'), encore que ce calcul est implicite, puisque A'=f[g($\underline{A_i}$)] et que g(A')=g($\underline{A_i}$).
- On opérerait de façon identique sur les parties de E si l'on avait n < m.
- L'avantage essentiel de cet algorithme est qu'il ne demande que très peu de place mémoire, puisqu'il n'utilise que le fermé précédent. Pour des tableaux relativement importants, c'est le seul qui soit utilisable.

2.3 *Méthode de Norris* (1978)

La méthode utilisée par M. Norris pour construire le treillis de Galois est l'application d'un principe très général qui s'applique à des problèmes d'algorithmique combinatoire très différents (construction des cliques maximales d'un graphe, énumération des monômes vides d'une algèbre de Boole, etc..). Pour simplifier la présentation nous considérerons que la relation binaire R entre E et F est donnée sous forme d'un tableau binaire comme pour l'exemple 1. Dans l'algorithme de Norris on considère ce tableau ligne à ligne et on construit l'ensemble des rectangles maximaux L^k obtenus après examen de k lignes, pour k variant de 1 à n. Initialement L^1 ne contient qu'un seul rectangle $x_1 x f(x_1)$. Une étape de cet algorithme revient à construire L^k connaissant x_k et L^{k-1}. Après examen des n lignes $\mathfrak{L} = L^n$.

A partir de x_k on construit L^k en considérant successivement tous les rectangles de L^{k-1}. Un rectangle X_i x Y_i est soit étendu en $(X_i \cup \{x_k\})$ x Y_i si Y_i est inclus dans $f(x_k)$, soit recopié et on ajoute à L^k le rectangle $(X_i \cup \{x_k\})$ x $(Y_i \cap f(x_k))$ si $Y_i \cap f(x_k)$ n'est pas la partie de F d'un rectangle de L^k. Enfin si après examen de tous les éléments de L^{k-1}, le rectangle x_k x $f(x_k)$ est maximum dans L^k, on l'y ajoute.

Algorithme de Norris
L := x_1 x $f(x_1)$
Pour k :=2 à n Faire
  Pour tout X_i x Y_i Œ L Faire
    Si $Y_i \subset f(x_k)$ Faire
      X_i x $Y_i \leftarrow (X_i \cup \{x_k\})$ x Y_i
      marquer x_k

Sinon Faire

Si $(X_i \cup \{x_k\}) \times (Y_i \cap f(x_k))$ est maximum Alors $L := L + (X_i \cup \{x_k\}) \times (Y_i \cap f(x_k))$

Fin de la boucle Pour tout $X_i \times Y_i$

Si x_k n'est pas marqué Alors $L := L + x_k \times f(x_k)$

Fin de la boucle k

EXEMPLE 4

Nous avons reporté en traitant l'exemple 1, suite à l'examen des lignes du tableau qui sont rappelées, chacune des listes de rectangles correspondant à chaque étape ; elles sont ici indicées, ce qui est inutile, comme il apparaît dans l'algorithme détaillé, où l'on ne traite qu'une seule liste L.

	a b c d e					
1 :	1 1 0 0 0	1	x	a,b		L^1
2 :	1 0 1 0 0	1	x	a,b		
		1,2	x	a		L^2
		2	x	a,c		
3 :	0 1 1 1 0	1	x	a,b		
		1,3	x	b		
		1,2	x	a		
		2	x	a,c		L^3
		2,3	x	c		
		3	x	b,c,d		
4 :	0 0 0 1 1	1	x	a,b		
		1,3	x	b		
		1,2	x	a		
		2	x	a,c		
		2,3	x	c		L^4
		3	x	b,c,d		
		3,4	x	d		
		4	x	d,e		
5 :	0 0 0 1 0	1	x	a,b		
		1,3	x	b		
		1,2	x	a		
		2	x	a,c		
		2,3	x	c		L^5
		3	x	b,c,d		
		3,4,5	x	d		
		4	x	d,e		
6 :	1 1 1 1 1	1,6	x	a,b		
		1,3,6	x	b		
		1,2,6	x	a		
		2,6	x	a,c		

2, 3,6	x	c	|	L^6
3,6	x	b,c,d	|	
3,4,5,6	x	d	|	
4,6	x	d,e	|	
6	x	a,b,c,d,e	|	

COMPLEXITÉ
Cet algorithme impose de conserver en mémoire la liste des rectangles en cours d'élaboration. Pour chaque rectangle, l'opération la plus longue est celle qui consiste à vérifier que $Y_i \cap f(x_k)$ n'est pas la partie de F d'un élément de L. Chaque étape de cet algorithme est de complexité $O(m.|L^k|^2)$.

2.4 *Méthode de Bordat* (1986)

Comme nous l'avons annoncé au début de ce paragraphe, l'algorithme de J.P. Bordat est le seul qui construise simultanément les éléments du treillis et les arêtes de son graphe de Hasse. Il engendre les rectangles maximaux $X'_i \times Y'_i$ *couverts* par un élément X x Y de $\mathfrak{L}$, c'est à dire sans intermédiaire entre X x Y et X' x Y' dans le treillis. Malheureusement un rectangle est engendré autant de fois qu'il a d'éléments immédiatement supérieurs; on passera donc un temps non négligeable dans cet algorithme à vérifier qu'un rectangle maximum que l'on vient d'engendrer n'est pas déjà calculé. Néanmoins cette opération permet d'énumérer toutes les arêtes du graphe de Hasse du treillis. Avant de détailler l'algorithme de J.P. Bordat nous rappelons ses définitions et son théorème que nous ne redémontrerons pas.

On dit que Y_i est une partie maximale de g(Y) x (F\Y) si et seulement si il n'existe pas de partie Z telle que : $Y_i \subset Z$ et $g(Y_i) \cap g(Y) = g(Z) \cap g(Y)$. La figure 3 ci-dessous montre trois parties maximales de g(Y) x (F\Y). Les rectangles encadrés sont supposés ne contenir que des 1 ; les 0 ne sont pas marqués.

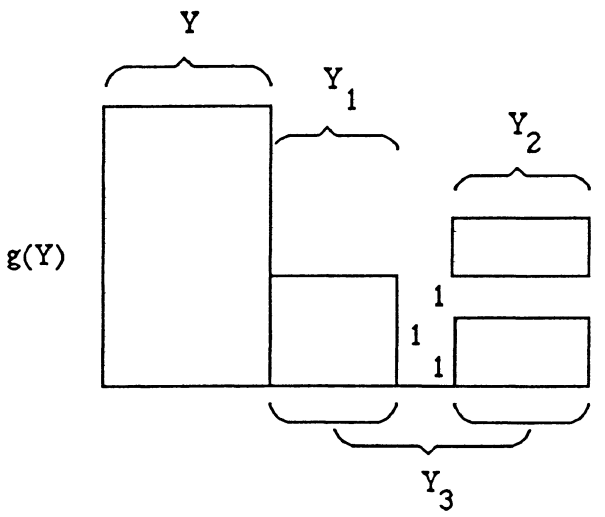

Figure 3

PROPOSITION 4. Le rectangle X x Y couvre $g(Y_i) \times (Y \cup Y_i)$ dans $\mathfrak{L}$ si et seulement si Y_i est une partie maximale de X x (F\Y).

L'algorithme de Bordat consiste à construire la liste des rectangles maximaux en partant du supremum du treillis E x ∅ (à moins qu'il existe des éléments de F liés à tous les éléments de E). Pour chaque élément de la liste, que l'on étudie séquentiellement, on construit les rectangles qu'il couvre. Pour chaque rectangle on cherche s'il n'est pas déjà contenu dans la liste. Sinon on prolonge la liste et dans tous les cas on ajoute un arc au graphe de Hasse. Quand le dernier élément de la liste a été examiné (il n'engendre que des arcs) l'algorithme s'arrête. Implémenté de cette manière, l'algorithme demande de conserver en mémoire la liste de tous les rectangles maximaux.

La description de J.P. Bordat est plus complexe, dans la mesure où il ne range pas les éléments du treillis dans une liste qu'il parcourt jusqu'à épuisement, mais dans un *arbre cousu* dans lequel un chemin constitue un fermé de Y. Nous ne reprenons pas ici cette structure de données, puisque nous nous intéressons aux principes des algorithmes et qu'il n'y a pas lieu de recopier ce qui est bien décrit et accessible dans la même revue. Nous détaillerons néanmoins une implémentation personnelle de la méthode pour construire les parties maximales de X x F\Y.

Soit $y_1, .. y_p$ les éléments de F\Y indicés dans l'ordre des sommes colonnes croissantes du tableau X x (F\Y), c'est à dire des cardinaux croissants des $g_X(y_i) = g(y_i) \cap X$. On construit un vecteur c de pointeurs tel que :

- $c(i) = 0$ si $g_X(y_i)$ est stictement inclus dans une autre colonne y_j,
- $c(i) = j$ si $g_X(y_i) = g_X(y_j)$
- $c(j) = -1$ si $g_X(y_i)$ est maximum et qu'il n'y a pas de colonne identique d'indice supérieur.

Ce vecteur c une fois construit en comparant $g_X(y_i)$ à $g_X(y_j)$ seulement pour $j > i$ puisque les colonnes sont indicées dans l'ordre des cardinaux croissants (complexité $O(|X| . |F\Y|^2)$), les parties maximales de X x (F\Y) se construisent en $O(|F\Y|)$.

Algorithme de Bordat

L := E x ∅
Pour tout X x Y ∈ L Faire
  Soit $y_1, .. y_p$ les éléments de F indicés dans l'ordre des cardinaux croissants des $g_X(y_i)$
  Pour tout i < j variant de 1 à p Faire
    Si $g(y_i) \subset$ (strictement) $g(y_j)$ Alors c(i) := 0
    Sinon Si $g(y_i) = g(y_j)$ Alors c(i) := j
      Sinon c(i) := -1
  Fin de la boucle pour tout i < j
  Pour i := 1 à p Faire
    Y' = ∅ ; j := i
    Tant que c(j) > 0 Faire
      Y' := Y' ∪ y_j
      k := j ; j := c(j) ; c(k) := 0
    Fin de Tant que
    Si c(j) = -1 et $g(Y') \times (Y \cup Y') \notin L$ Alors L := L + $g(Y') \times (Y \cup Y')$
  Fin de la boucle i
Fin de la boucle Pour tout X x Y

EXEMPLE 5

Dans cette application de l'algorithme à la correspondance de l'exemple 1, nous avons numéroté les rectangles maximaux dans leur ordre d'apparition. Pour chacun nous avons construit le sous-tableau X x (F\Y) (sans réordonner les colonnes) et listé les parties maximales Y_i. Chaque rectangle $g(Y_i)$ x $(Y \cup Y_i)$ est un élément du treillis de Galois dont on a reporté le numéro d'élément au dessous. On en déduit la liste des arcs du graphe de Hasse : (1,2), (1,3), (1,4), (1,5), (2,6), (2,7), (3,6), (3,8), (4,7), (5,8), (5,9), (6,10), (7,10), (8,10), (9,10).

1 = 1,2,3,4,5,6 x ∅

	a	b	c	d	e	
1	1	1				
2	1		1			
3		1	1	1		
4				1	1	
5				1		{a},{b},{c},{d} maximum
6	1	1	1	1	1	2 3 4 5

2 = 1, 2, 6 x a

	b	c	d	e	
1	1				
2		1			{b},{c} maximum
6	1	1	1	1	6 7

3 = 1, 3, 6 x b

	a	c	d	e	
1	1				
3		1	1		{a},{c,d} maximum
6	1	1	1	1	6 8

4 = 2, 3, 6 x c

	a	b	d	e	
2	1				
3		1	1		{a},{b,d} maximum
6	1	1	1	1	7 8

5 = 3, 4, 5, 6 x d

	a	b	c	e	
3		1	1		
4				1	
5					{b,c},{e} maximum
6	1	1	1	1	8 9

6 = 1, 6 x a, b

	c	d	e	
1				{c,d,e} maximum
6	1	1	1	10

7 = 2, 6 x a, c

	b	d	e	
2				{b,d,e} maximum
6	1	1	1	10

8 = 3, 6 x b, c, d

	a	e	
3			{a,e} maximum
6	1	1	10

9 = 4, 6 x d, e

	a	b	c	
4				{a,b,c} maximum
6	1	1	1	10

10 = 6 x a, b, c, d, e

COMPLEXITÉ

On peut facilement, au cours de cet algorithme, calculer une fonction rang qui détermine dans le tracé du graphe de Hasse le niveau auquel doit être placé chaque sommet. La fonction rang, initialisée à 0 et définie par :

$$rang(X'xY') = Sup\ (\ rang(X'xY'), rang(XxY)+1\ )$$

laisse le sommet 1 au niveau 0 et place pour l'exemple précédent les sommets 2, 3, 4, 5 au niveau 1, les sommets 6, 7, 8, 9 au niveau 2 et le sommet 10 au niveau 3.

3. CONCLUSION

Une comparaison de ces méthodes sur des bases théoriques n'est pas simple. Nous avons évalué la complexité des étapes de chaque algorithme ; leur nombre n'est pas toujours prévisible. Sur ce point nous ne pouvons que répéter que l'algorithme de Ganter est le seul utilisable pour des correspondances portant sur des ensembles de plus de, disons, 15 éléments. La méthode de Bordat est la seule qui construit directement le graphe de Hasse du treillis.

Sur des bases expérimentales, nous avons réalisé des comparaisons de manière empirique en programmant les quatre méthodes. La structure de données commune est très simple. Chaque rectangle est codé dans une ligne d'un tableau X (à n colonnes, pour la partie de E) et une ligne d'un tableau Y (à m colonnes, pour la partie de F). X contient les numéros des lignes de la table qui code la correspondance et Y un codage booléen des colonnes. On cherche souvent si un rectangle est déjà présent dans la liste L (les lignes de ces deux tableaux). On utilise pour cela, avant d'effectuer des comparaisons exhaustives, soit le nombre d'éléments de X soit la valeur entière dont Y est la numération binaire. A partir de nombreux tirages aléatoires de correspondances, nous avons relevé les temps nécessaires. Plutôt que de transcrire des moyennes, nous vous faisons partager nos conclusions.

La méthode de Norris est légèrement plus efficace que celle de Ganter. Celle de Bordat serait, disons, deux fois plus longue que celle de Norris et deux fois plus rapide que celle de Chein. Les algorithmes détaillés ci-dessus permettrons au lecteur intéressé par la programmation de se faire une idée plus précise.

BIBLIOGRAPHIE

BARBUT M., MONJARDET B., *Ordre et Classification*, Algèbre et Combinatoire (tome 2), Paris, Hachette, 1970.

BIRKHOFF G., *Lattice Theory*, A.M.S., 25, Providence, 1967.

BORDAT J.P., Calcul pratique du treillis de Galois d'une correspondance, *Mathématiques et Sciences humaines*, 96, 1986, p. 31-47.

CHEIN M., Algorithme de recherche des sous-matrices premières d'une matrice, *Bull. Math. R.S. Roumanie*, 13, 1969.

DUQUENNE V., GUIGUES J.L., Familles minimales d'implications informatives résultant d'un tableau de données binaires, *Mathématiques et Sciences humaines*, 95, 1986, p. 5-18.

FAY G., An algorithm for finite Galois connexions, *Journal of Computational Linguistic and Computational Languages*, 10, 1975, p. 99-123.

GANTER B., *Two basic algorithms in Concept Analysis*, Preprint 831, Technische Hochschule Darmstadt, 1984, 28p.

KAUFMANN A., Pichat E., *Méthodes mathématiques non numériques et leurs algorithmes*, Paris, Masson, 1977.

LAVALLÉE I., *Rectangles maximaux dans une relation binaire quelconque*, Preprint Université Paris VI, p. 157-171.

MALGRANGE Y., Recherche des sous-matrices premières d'une matrice à coefficients binaires; Application à certains problèmes de graphe, *Deuxième congrès de l'AFCALTI*, Gauthier-Villars, 1962, p. 231-242.

NORRIS E.M., An algorithm for computing the maximal rectangles in a binary relation, *Revue Roumaine de Mathématiques Pures et Appliquées*, 1978, 23, 2, p. 243-250.

WILLE R., Restructuring lattice theory : an approach based on hierarchies of concepts, in *Ordered Sets*, I. Rival (Ed.), Dordrecht, Reidel, 1982.

WILLE R., Knowledge acquisition by methods of formal concept analysis, *Data Analysis, Learning symbolic and numeric knowledge*, E. Diday (Ed.), New York, Nova Science Publisher, 1989, p. 365-380.