Contribution à l'algorithmique non numérique dans les ensembles ordonnés

Etienne Pichat

N° d'ordre

THESES

présentées à

LA FACULTE DES SCIENCES DE L'UNIVERSITE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR ES SCIENCES MATHEMATIQUES

par

Etienne Pichat

1ère thèse :

Contribution à l'algorithmique non numérique dans les ensembles ordonnés

2ème thèse :

PROPOSITIONS DONNEES PAR LA FACULTE

Thèses soutenues le 17 Octobre 1970, devant la commission d'examen

Monsieur J. KUNTZMANN	Président
Monsieur FORTET	Invité
Monsieur C. BENZAKEN	Examinateur
Monsieur M. GRIFFITHS	Examinateur

L I S T E D E S P R O F E S S E U R S

Doyen honoraire : Monsieur M. MORET
Doyen : Monsieur E. BONNIER

PROFESSEURS TITULAIRES

MM.	NEEL Louis	Physique Expérimentale
	KRAVTCHENKO Julien	Mécanique Rationnelle
	CHABAUTY Claude	Calcul différentiel et intégral
	BENOIT Jean	Radioélectricité
	CHENE Marcel	Chimie Papetière
	FELICI Noël	Electrostatique
	KUNTZMANN Jean	Mathématiques Appliquées
	BARBIER Reynold	Géologie Appliquée
	SANTON Lucien	Mécanique des Fluides
	OZENDA Paul	Botanique
	FALLOT Maurice	Physique Industrielle
	KOSZUL Jean-Louis	Mathématiques
	GALVANI Octave	Mathématiques
	MOUSSA André	Chimie Nucléaire
	TRAYNARD Philippe	Chimie Générale
	SOUTIF Michel	Physique Générale
	CRAYA Antoine	Hydrodynamique
	REULOS René	Théorie des Champs
	BESSON Jean	Chimie Minérale
	AYANT Yves	Physique Approfondie
	GALLISSOT François	Mathématiques
Melle.	LUTZ Elisabeth	Mathématiques
MM.	BLAMBERT Maurice	Mathématiques
	BOUCHEZ Robert	Physique Nucléaire
	LLIBOUTRY Louis	Géophysique
	MICHEL Robert	Minéralogie et pétrographie
	BONNIER Etienne	Electrochimie et Electrométallurgie
	DESSAUX Georges	Physiologie animale
	PILLET Emile	Physique Industrielle-Electrotechnique
	YOCCOZ Jean	Physique Nucléaire théorique
	DEBELMAS Jacques	Géologie Générale
	GERBER Robert	Mathématiques
	PAUTHENET René	Electrotechnique
	MALGRANGE Bernard	Mathématiques Pures
	VAUQUOIS Bernard	Calcul Electronique
	BARJON Robert	Physique Nucléaire

MM.	BARBIER Jean-Claude	Physique
	SILBER Robert	Mécanique des Fluides
	BUYLE-BODIN Maurice	Electronique
	DREYFUS Bernard	Thermodynamique
	KLEIN Joseph	Mathématiques
	VAILLANT François	Zoologie et Hydrobiologie
	ARNAUD Paul	Chimie
	SENGEL Philippe	Zoologie
	BARNOUD Fernand	Biosynthèse de la Cellulose
	BRISSONNEAU Pierre	Physique
	GAGNAIRE Didier	Chimie Physique
Mme.	KOFLER Lucie	Botanique
MM.	DEGRANGE Charles	Zoologie
	PEBAY-PEROULA Jean-Claude	Physique
	RASSAT André	Chimie Systématique
	DUCROS Pierre	Cristallographie Physique
	DODU Jacques	Mécanique Appliquée I. U. T.
	ANGLES D'AURIAC Paul	Mécanique des Fluides
	LACAZE Albert	Thermodynamique
	GASTINEL Noël	Analyse numérique
	GIRAUD Pierre	Géologie
	PERRET René	Servo-mécanisme
	PAYAN Jean-Jacques	Mathématiques Pures

PROFESSEURS SANS CHAIRE

MM.	GIDON Paul	Géologie
Mme.	BARBIER Marie-Jeanne	Electrochimie
Mme.	SOUTIF Jeanne	Physique
	COHEN Joseph	Electrotechnique
	DEPASSEL R.	Mécanique des Fluides
	GLENAT René	Chimie
	BARRA Jean	Mathématiques Appliquées
	COUMES André	Electronique
	PERRIAUX Jacques	Géologie et Minéralogie
	ROBERT André	Chimie Papetière
	BIARREZ Jean	Mécanique Physique
	BONNET Georges	Electronique
	CAUQUIS Georges	Chimie Générale
	BONNETAIN Lucien	Chimie Minérale
	DEPOMIER Pierre	Physique Nucléaire-Génie Atomique
	HACQUES Gérard	Calcul numérique
	POLOUJADOFF Michel	Electrotechnique
Mme.	KAHANE Josette	Physique
Mme.	BONNIER Jane	Chimie
MM.	VALENTIN Jacques	Physique
	REBECQ Jacques	Biologie
	DEPORTES Charles	Chimie
	SARROT-REYNAULD Jean	Géologie
	BERTRANDIAS Jean-Paul	Mathématiques Appliquées
	AUBERT Guy	Physique

PROFESSEURS ASSOCIES

MM.	RODRIGUES Alexandre	Mathématiques Pures
	MORITA Susumu	Physique Nucléaire
	RADHAKRISHNA	Thermodynamique

MAITRES DE CONFERENCES

MM.	LANCIA Roland	Physique Atomique
Mme.	BOUCHE Liane	Mathématiques
MM.	KAHANE André	Physique Générale
	DOLIQUE Jean Michel	Electronique
	BRIERE Georges	Physique
	DESRE Georges	Chimie
	LAJZEHOWICZ Joseph	Physique
	LAURENT Pierre	Mathématiques Appliquées
Mme.	BERTRANDIAS Françoise	Mathématiques Pures
MM.	LONGEQUEUE Jean-Pierre	Physique
	SOHM Jean-Claude	Electrochimie
	ZADWORNY François	Electronique
	DURAND Francis	Chimie Physique
	CARLIER Georges	Biologie végétale
	PFISTER Jean-Claude	Physique
	CHIBON Pierre	Biologie animale
	IDELMAN Simon	Physiologie animale
	BLOCH Daniel	Electrotechnique I. P.
	MARTIN-BOUYER Michel	Chimie (C. S. U. Chambéry)
	SIBILLE Robert	Construction mécanique (I. U. T.)
	BRUGEL Lucien	Energétique I. U. T.
	BOUVARD Maurice	HYdrologie
	RICHARD Lucien	Botanique
	PELMONT Jean	Physiologie animale
	BOUSSARD Jean-Claude	Mathématiques Appliquées (I. P. G.)
	MOREAU René	Hydraulique I. P. G.
	ARMAND Yves	Chimie I. U. T.
	BOLLIET Louis	Informatique I. U. T.
	KUHN Gérard	Energétique I. U. T.
	PEFFEN René	Chimie I. U. T.
	GERMAIN Jean-Pierre	Mécanique
	JOLY Jean-René	Mathématiques Pures
Melle.	PIERY Yvette	Biologie animale
	BERNARD Alain	Mathématiques Pures
	MOHSEN Tahsin	Biologie (C. S. U. Chambéry)
	CONTE René	Mesures Physiques I. U. T.
	LE JUNTER Noël	Génie Electrique Electronique I. U. T.
	LE ROY Philippe	Génie Mécanique I. U. T.
	ROMIER Guy	Techniques Statistiques quantitatives I. U. T.
	VIALON Pierre	Géologie
	BENZAKEN Claude	Mathématiques Appliquées
	MAYNARD Roger	Physique

MM.	DUSSAUD René	Mathématiques (C. S. U. Chambéry)
	BELORIZKY Elie	Physique (C. S. U. Chambéry)
Mme.	LAJZEROWICZ Jeannine	Physique (C. S. U. Chambéry)
M.	JULLIEN Pierre	Mathématiques Pures
Mme.	RINAUDO Marguerite	Chimie
MM.	BLIMAN Samuel	E. I. E.
	BEGUIN Claude	Chimie Organique
	NEGRE Robert	I. U. T.

MAITRES DE CONFERENCES ASSOCIES

MM.	YAMADA Osamu	Physique du Solide
	NAGAO Makoto	Mathématiques Appliquées
	MAREZIO Massimo	Physique du Solide
	CHEECKE John	Thermodynamique
	BOUDOURIS Georges	Radioélectricité
	ROZMARIN Georges	Chimie Papetière

Je voudrais exprimer toute ma reconnaissance à Monsieur le Professeur KUNTZMANN, Directeur de l'Institut de Mathématiques Appliquées de Grenoble, pour m'avoir donné la possibilité d'effectuer cette étude, pour les précieuses indications qu'il n'a cessé de me donner dans la poursuite de ce travail, et pour l'honneur qu'il me fait en acceptant de présider le Jury.

Que Monsieur FORTET, Professeur à la Sorbonne, me permette de lui adresser le témoignage de ma respectueuse gratitude pour l'intérêt qu'il a bien voulu porter à ce travail et pour avoir accepté de participer au Jury.

Je remercie vivement Monsieur BENZAKEN, Maître de Conférences, pour les encouragements qu'il m'a prodigués.

Monsieur GRIFFITHS, Maître de Conférences, m'a permis en me donnant le sujet d'une deuxième thèse, d'élargir mes connaissances ; qu'il reçoive l'expression de ma grande reconnaissance.

Je tiens aussi à remercier Mademoiselle PAYERNE pour le soin efficace apporté dans la dactylographie de cette thèse.

INTRODUCTION

Cette thèse est une contribution à l'algorithmique combinatoire.

Son point de départ est le problème fondamental de l'algèbre de Boole : la représentation minimale des fonctions booléennes.

Si la recherche d'une représentation minimale sous forme de sommes de monômes ne présente pas de difficulté de principe, par contre son temps d'exécution augmente beaucoup plus rapidement que le nombre de variables. Une approche préliminaire de la synthèse des fonctions booléennes est donc de les décomposer en fonctions de fonctions ; cette méthode est approfondie et généralisée aux fonctions quelconques au cours des deux premiers chapitres.

Dans les chapitres 3 à 7, on considère que la détermination de l'ensemble des monômes premiers et celle d'une couverture minimale ne sont que des cas particuliers du problème de la recherche des éléments maximaux de structures algébriques partiellement ordonnées dont les opérations algébriques sont isotones et en nombre fini.

Une telle généralisation a le mérite de relier et d'unifier un grand nombre de problèmes combinatoires (en théorie des graphes, des relations, des ensembles ordonnés...). Elle nous a permis de définir des algorithmes généraux et d'expliquer dans ce cadre plus vaste l'origine des performances de cas particuliers et a priori éloignés les uns des autres. Cette théorie effectue donc une synthèse, mais elle fournit aussi un cadre propice à la comparaison, à l'amélioration et à la conception des algorithmes.

Le chapitre 1 prolonge les travaux de PICHAT [1] sur les décompositions simples des fonctions booléennes.

D'abord, les rapports entre décompositions et synthèse minimale des fonctions booléennes sont étudiés.

Ensuite, on énonce quelques résultats concernant des fonctions particulières décomposables disjonctivement et on énumère les fonctions booléennes de quatre variables admettant une décomposition disjointe simple.

Enfin, est mentionnée la réalisation avec DESCHIZEAUX d'un algorithme très commode de détermination des décompositions simples d'une fonction booléenne à partir de son écriture galoisienne : il se réduit à un calcul sur des partitions de variables, indépendamment des fonctions booléennes associées.

Le chapitre 2 étudie la structure algébrique de l'ensemble des décompositions des fonctions de plusieurs variables quelconques.

On étend d'abord au cas de décompositions non disjointes et de variables prenant un nombre infini de valeurs, les résultats obtenus par KARP dans le cas de variables finies.

Est introduite, ensuite, la notion de décomposition injective ; elle permet de déterminer dans quelles conditions l'existence de deux décompositions en chaîne entraîne l'existence d'une décomposition plus fine.

Enfin, est défini le concept de dépendance de cardinal n d'une fonction par rapport à un sous-ensemble A de ses variables ; la notion classique d'indépendance d'une fonction par rapport à A est alors équivalente à une dépendance de cardinal un. Dans le cas où les variables prennent un nombre fini de valeurs, les résultats obtenus précédemment s'expriment très simplement.

En résumé, on répond aux questions que les propriétés des décompositions booléennes, en particulier leur structure de treillis mise en avant par ASHENHURST, conduisent à se poser : à quoi tient l'existence de propriétés si remarquables ? Sont-elles généralisables ?

Le but du chapitre 3 est d'étudier les éléments appelés pavés du produit cardinal G de treillis distributifs G_1, G_2, ..., G_I, principalement de déterminer les pavés compatibles avec une partie finie A de G (cette notion de compatibilité est précisée dans la première partie de ce chapitre) : dans le cas où chaque

treillis G_i est le treillis de Boole à quatre éléments, ce problème est celui de la recherche des monômes premiers d'une fonction booléenne ; dans le cas où G est le produit de deux treillis booléens, c'est le problème de la détermination des rectangles maximaux inclus dans une partie d'un rectangle.

L'algorithme de sélection, donné dans la deuxième partie généralise et résoud la conjecture de TISON. Pour cela on autorise l'adjonction en cours d'algorithme de pavés compatibles avec A ; la restriction à un pavé devient un homomorphisme relativement au processus de sélection ; un raisonnement par induction achève la démonstration.

La troisième partie fournit un autre algorithme pour connaître si une réunion de pavés couvre ou non le plus grand élément de G et généralise la notion de fonction booléenne aux treillis distributifs.

Dans le chapitre 4, on présente le problème de la recherche de tous les éléments maximaux d'une structure algébrique $G = \langle A, \leq, * \rangle$ finiment engendré par A (condition de finitude), partiellement ordonnée par $\leq$ et dont les opérations algébriques $*$ sont isotones et en nombre fini.

On propose ensuite trois algorithmes en file pour le traiter. On montre que si G satisfait une certaine condition générale (par exemple lorsque ses opérations $*$ sont distributives), ses éléments maximaux peuvent être obtenus par un algorithme moins primitif, appelé algorithme C, itération d'un quelconque des trois algorithmes précédents.

Ces algorithmes en file sont exhaustivement combinatoires. Or les structures $\langle A, \leq, * \rangle$ peuvent avoir des propriétés telles que certaines combinaisons d'éléments sont a priori inutiles (une formulation théorique, utilisant la notion de fonction de localisation, en est donnée) ; il est donc intéressant de trouver des dispositifs permettant d'appliquer les algorithmes précédents dans une version simplifiée. Deux supports s'avèrent particulièrement efficaces pour éviter ces combinaisons inutiles : une disposition matricielle et une disposition en produit d'ensembles qui donnent respectivement naissance aux algorithmes dits matriciels et aux développements ; ils sont l'objet des chapitres 6 et 7.

Le chapitre 5 est l'illustration des quatre algorithmes en file précédents dans les deux cas suivants :

- la recherche des pavés maximaux compatibles avec une partie finie A ; on obtient des algorithmes de BENZAKEN, CHEIN, GILLI et MEO, KUNTZMANN, MALGRANGE, QUINE, TISON ; des applications sont données ;

- la détermination de la relation d'équivalence réunion de relations d'équivalences définies sur un ensemble fini, problème pour lequel un algorithme plus élaboré est aussi proposé.

Le chapitre 6 étudie les algorithmes matriciels. Il donne en particulier l'algorithme CM, une version matricielle de l'algorithme C ayant le mérite de préciser, quand A^* ne vérifie pas la condition de finitude, les éléments maximaux en nombre infini ; cet algorithme est irredondant et très efficace.

Il applique ensuite les algorithmes matriciels en particuler à l'énumération de chemins dans les graphes finis, ce qui permet de retrouver des algorithmes de DANTZIG, KAUFMANN, McNAUGHTON et YAMADA, PAIR et DERNIAME.

Enfin, le problème de la fermeture transitive d'une matrice carrée M à éléments dans un gerbier est examiné ; il est ramené à celui de la recherche des éléments maximaux d'un monoïde $\langle A(M), \leq, * \rangle$ associé à M. Les résultats de McNAUGHTON et YAMADA relatifs aux matrices d'expressions régulières sont généralisés, ceux de BELLMAN et KALABA, BENZAKEN, FORD, FOULKES, KUNTZMANN, LUNC, MAGHOUT, MALGRANGE, ROBERT-FERLAND, ROY, TOMESCU, WARSHALL... concernant la fermeture des matrices définies sur des pseudo-trèillis distributifs sont retrouvés.

Le chapitre 7 traite des développements. On en donne d'abord la définition, on montre que ce sont des algorithmes particuliers de recherche d'éléments maximaux, et on en distingue deux types : les développements progressifs et les développements différenciés ou sériels.

A titre d'exemples de développement, sont examinés l'énumération des chemins de longueur donnée dans les graphes comme l'ont faite KAUFMANN, MAGHOUT, PAIR et DERNIAME, l'énumération des chemins et circuits hamiltoniens comme l'a faite BOUCHET, l'algorithme de McCLUSKEY donnant les monômes premiers d'une fonction booléenne, l'énumération des ensembles couplants maximaux d'un graphe simple.

On donne ensuite un nouvel algorithme pour trouver les pavés maximaux compatibles avec une partie finie du produit de treillis distributifs : un double développement de produits de réunions de pavés ; c'est l'extension d'un algorithme

booléen de KUNTZMANN. Dans le cas de treillis booléens, un algorithme par double complémentation est aussi possible ; application en est faite à la détermination des ensembles fortement stables et des cliques maximaux d'un hypergraphe : on retrouve des résultats de BENZAKEN et de MAGHOUT.

La dernière partie traite du cas particulier très courant des couvertures minimales ou développements de produits de sommes de lettres. Il y est donné principalement :

- un algorithme différencié lexicographique construisant chaque couverture minimale une fois et une seule ; son application aux produits de sommes de deux lettres s'avère rapide sur ordinateurs ; ARSAC et KUNTZMANN ont utilisé un procédé similaire pour déterminer les ensembles extérieurement stables maximaux d'un graphe ;

- un développement progressif des produits de sommes de deux lettres alliant en particulier les avantages d'une part de celui de BEDNAREK et TAULBEE, d'autre part de celui de BENZAKEN.

TABLE DES MATIERES

CHAPITRE 1

SUR LES DECOMPOSITIONS DES FONCTIONS BOOLEENNES

I - DECOMPOSITIONS ET ECRITURES MINIMALES DES FONCTIONS BOOLEENNES [P3,4].

I-1. Introduction et définitions.

Nous nous intéressons ici à la minimisation du nombre total de lettres des représentations à l'aide des opérateurs somme (+), produit (.) et négation (') d'une fonction booléenne. Une telle réalisation minimale d'une fonction booléenne quelconque non seulement existe toujours mais n'est pas généralement unique. En algèbre de Boole, comme en théorie des réseaux électriques et des réseaux de contact [S,R] [L,C], la minimisation est une question non résolue : aucune caractérisation de la famille d'écritures minimales réalisant une fonction booléenne donnée n'est connue jusqu'à présent, les contributions au problème de la réalisation minimale étant restreintes à des classes limitées de fonctions [K].

Après un rappel des définitions employées, nous donnerons d'abord une méthode de synthèse des écritures minimales d'une fonction booléenne complète ou incomplète à l'aide des décompositions, puis nous nous intéresserons aux écritures minimales d'une classe particulière de fonctions : les fonctions admettant une décomposition disjointe simple.

Nous appellerons écriture une écriture utilisant les opérateurs somme, produit et négation, coût d'une écriture le nombre (d'occurrences) de ses lettres (accentuées ou non), coût d'une fonction booléenne f(X) (que nous noterons [f(X)]) le coût de son ou de ses écritures ayant les plus faibles coûts et écriture minimale de f(X) tout écriture de f(X) de coût [f(X)].

I-2. Synthèse des écritures minimales d'une fonction booléenne.

a. Rappels et cas général.

La recherche d'une écriture minimale de la fonction booléenne incomplète $\varphi(X)$ peut être ramenée à la recherche d'une écriture croissante minimale de la fonction incomplète croissante $\varphi^o(X,X^o)$ attachée à $\varphi(X)$ (la définition en est donnée dans [K] p. 192) : en effet, une écriture de $\varphi(X)$ donne en remplaçant X' par X^o une expression croissante compatible avec $\varphi^o(X,X^o)$ et réciproquement, une écriture booléenne croissante de $\varphi^o(X,X^o)$ donne, en remplaçant X^o par X', une écriture compatible avec $\varphi(X)$ (Voir [K] p. 331).

Or la détermination d'une écriture croissante minimale peut se faire à l'aide de celle des décompositions simples additives et du principe de dualité ou, plus trivialement, en comptant les lettres répétées ; nous avons déjà utilisé cette remarque dans [P1], nous le ferons encore dans l'exemple de b.

Enfin, la recherche des décompositions simples additives est facilitée par la propriété suivante [P1,2] : une condition nécessaire pour qu'une fonction incomplète ait une fonction compatible croissante admettant une décomposition simple additive suivant la partition (A, B, Z) est que la somme des monômes premiers de sa borne supérieure uniques diviseurs de monômes premiers de sa borne inférieure admette une décomposition simple additive suivant la partition (A, B, Z).

b. Cas particulier de la synthèse d'une écriture minimale des fonctions privilégiées.

Rappelons qu'une fonction booléenne est privilégiée si elle possède une écriture utilisant une seule fois toute variable par rapport à laquelle elle est monotone et deux fois les autres variables (sous forme directe et sous forme complémentée) ; c'est donc un réseau à double contact [C,L] et série parallèle.

Le paragraphe précédent permet d'affirmer qu'une condition nécessaire et suffisante pour qu'une fonction soit privilégiée est qu'une fonction privilégiée croissante soit compatible avec sa fonction croissante attachée. La recherche d'une fonction privilégiée croissante va être illustrée à l'aide de l'exemple qui suit.

Exemple 1

$$h(x,y,z,t) = xyzt + x'(y' + z' + t') + z't' \qquad (1)$$

n'est pas une fonction privilégiée. En effet, il suffit de montrer que sa fonction croissante attachée $h^o(x,x^o,y,y^o,z,z^o,t,t^o)$ définie par :

$$\underline{h^o} = (xyzt + x^oy^o + x^oz^o + x^ot^o + z^ot^o)(x + x^o)(y + y^o)(z + z^o)(t + t^o)$$

$$= xyzt + x^oy^ozt + x^oyz^ot + x^oyzt^o + xyz^ot^o + x^oy^oz^ot + x^oy^ozt^o + x^oyz^ot^o + xy^oz^ot^o + x^oy^oz^ot^o$$

$$\bar{h}^o = xyzt + x^oy^o + x^oz^o + x^ot^o + z^ot^o + xx^o + yy^o + zz^o + tt^o$$

n'admet pas d'écriture croissante privilégiée :

1. $h^o(x,x^o,y,y^o,z,z^o,t,t^o)$ n'admet pas de décomposition disjointe additive.

En effet, la somme des monômes premiers de $\bar{h}^o$ uniques diviseurs de monômes premiers de $\underline{h^o}$, $xyzt + x^oy^o + x^oz^o + x^ot^o + z^ot^o$, montre que $(xyzt, x^oy^oz^ot^o)$ serait la seule partition possible pour une telle décomposition ; la fonction $h_1(x^o,y^o,z^o,t^o)$ de la décomposition additive correspondant de $h^o = h_1 + h_2\ (x,y,z,t)$

$$h_1(x^o,y^o,z^o,t^o) = x^oy^o + x^oz^o + x^ot^o + z^ot^o$$

n'admet pas d'écriture croissante privilégiée puisque $h_1(x^o,0,z^o,t^o)=x^oz^o+x^ot^o+z^ot^o$.

2. $h^o(x,x^o,y,y^o,z,z^o,t,t^o)$ n'admet pas de décomposition disjointe multiplicative car sa duale $h^{o*}(x,x^o,y,y^o,z,z^o,t,t^o)$ définie par :

$$\underline{h}^{o*} = (\bar{h}^o)^* = xy^oz^ot^o+x^oyzt^o+x^oyz^ot+x^oyz^ot^o+x^oy^ozt^o+x^oy^oz^ot$$

$$\bar{h}^{o*} = (\underline{h}^o)^* = xy^oz^ot^o+x^oyz^o+x^oyt^o+x^ozt^o+x^oz^ot+xx^o+yy^o+zz^o+tt^o$$

n'en admet pas qui soit additive. En effet, la somme des monômes premiers de $\bar{h}^{o*}$ uniques diviseurs de monômes premiers de $\underline{h}^{o*}$ est : $xy^oz^ot^o + x^ozt^o + x^oz^ot$.

On a démontré que :

$[h(x,y,z,t)] \geq 9$.

Avec la même méthode, on peut même démontrer que

$[h(x,y,z,t)] \geq 10$,

c'est-à-dire que (1) est une écriture minimale.

c. Ce procédé de synthèse pour les écritures minimales d'une fonction quelconque est évidemment efficace quand le nombre de variables est peu élevé ou pas trop inférieur au coût de la fonction ; il est préférable que son utilisation aille de pair avec l'exploitation des propriétés particulières des décompositions disjointes simples (voir [K,L,P] et le paragraphe I-3 suivant).

I-3. Ecritures minimales de fonctions admettant une décomposition disjointe simple.

a. Si une fonction admet une décomposition disjointe simple suivant la partition (A,B) de ses variables, toutes ses écritures minimales peuvent

ne pas admettre de décomposition disjointe simple suivant (A,B), comme le montrent les deux exemples suivants. Remarquons d'abord que $a \oplus h = ah' + a'h$, monotone ni en a, ni en h, admet pour écriture minimale : $ah' + a'h$.

Exemple 2.

$$a \oplus b \oplus h(C)$$

s'écrit, en utilisant sa décomposition disjointe simple suivant la partition (ab,C) :

$$(a \oplus b)\, h'(C) + (a \oplus b)'\, h(C) = (ab' + a'b)\, h'(C) + (ab + a'b')\, h(C)$$

de coût $8 + 2[h(C)]$ et, en utilisant sa décomposition suivant (a,bC) :

$$a[b \oplus h(C)]' + a'[b \oplus h(C)] = a[bh(C) + b'h'(C)] + a'[bh'(C) + b'h(C)]$$

de coût $6 + 4[h(C)]$.

Plus explicitement, nous pouvons dresser le tableau suivant donnant le coût de $a \oplus b \oplus h(C)$ en fonction du coût de h et de la décomposition choisie :

[a⊕b⊕h(C)]	1	2	3	4	...	[h(C)]
décomposition suivant (ab,C)	10	12	14	16	...	8+2[h(C)]
décomposition suivant (a,bC)	10	14	18	22	...	6+4[h(C)]

Donc si $[h(C)] \geq 2$, l'utilisation de la décomposition suivant (a,bC) conduit à une écriture non minimale.

Donnons maintenant un exemple plus complexe, mais ne faisant intervenir qu'une seule décomposition disjointe simple et ne pouvant donc laisser croire à l'existence de "bonnes" décompositions parmi des "moins bonnes" :

Exemple 3.

Considérons la fonction

$$f(a,x,y,z,t)) = g[a,h(x,y,z,t)] = a \oplus h(x,y,z,t),$$

où $h(x,y,z,t) = xyzt + x'(y' + z' + t') + z't'$,
et montrons qu'une écriture de $g(a,h)$, où h est affecté du poids $[h(x,y,z,t)]$, ne peut pas être une écriture minimale de $f(a,x,y,z,t)$.

L'écriture suivante de $f(a,x,y,z,t)$

$f(a,x,y,z,t) = (a'x + ax')\ yzt + [a'x' + ax(z + t)]\ (y'+z'+t') + a'z't'$

a un coût de 19 ; d'où :

$$[f(a,x,y,z,t)] \leq 19$$

Or une écriture minimale de $g(a,h) = a \oplus h$ contenant a,a',h,h' et le coût de $h(x,y,z,t)$ étant, d'après l'exemple 1, supérieur ou égal à 9, on a :

$$[g[a,h(x,y,z,t)]] \geq 20.$$

Etudions maintenant des décompositions disjointes simples particulières dont l'existence entraîne celle d'une écriture minimale décomposable.

b. Donnons d'abord une généralisation aux fonctions incomplètes du théorème 1 de [B] :

Théorème 1.

Soit une fonction booléenne incomplète $\varphi(A,B)$ dont les bornes inférieure et supérieure admettent des décompositions disjointes simples par rapport à l'opérateur $*$(+ ou .) suivant la partition (A,B) de ses variables. Alors l'une au moins de ses écritures minimales admet une décomposition disjointe simple par rapport à l'opérateur $*$ suivant la partition (A,B).

Démonstration.

Supposons que l'opérateur $*$ est l'opérateur produit (le cas où $*$ est $+$ est dual du cas étudié ; nous ne nous en préoccuperons donc pas) et posons :

$$\begin{cases} \underline{\varphi}(A,B) = \underline{h}(A) \,.\, \underline{k}(B) \\ \overline{\varphi}(A,B) = \overline{h}(A) \,.\, \overline{k}(B) \end{cases}$$

Alors $\underline{\varphi}(A,B) \leq \overline{\varphi}(A,B)$ entraîne :

$$\begin{cases} \underline{h}(A) \leq \overline{h}(A) \\ \underline{k}(B) \leq \overline{k}(B), \end{cases}$$

inégalités qui justifient les notations $\underline{h}(A)$ et $\overline{h}(A)$, $\underline{k}(B)$ et $\overline{k}(B)$.

Considérons une écriture minimale $f(A,B)$ de $\varphi(A,B)$:

$$\underline{\varphi}(A,B) \leq f(A,B) \leq \overline{\varphi}(A,B)$$

ou : $$\underline{h}(A) \,.\, \underline{k}(B) \leq f(A,B) \leq \overline{h}(A) \,.\, \overline{k}(B) \qquad (1)$$

et soit B_o une configuration des variables B telle que $\underline{k}(B_o) = 1$. Alors $\overline{k}(B_o) = 1$ et (1) devient :

$$\underline{h}(A) \leq f(A,B_o) \leq \overline{h}(A).$$

Semblablement, pour A_o vérifiant : $\underline{h}(A_o) = 1$, (1) devient :

$$\underline{k}(B) \leq f(A_o,B) \leq \overline{k}(B).$$

$f(A,B_o) \,.\, f(A_o,B)$ est de coût au plus égal à celui de $f(A,B)$ et est compatible avec $\varphi(A,B)$.

c. Théorème 2.

Soit une fonction incomplète $\varphi(A,B)$ dont les bornes inférieure et supérieure admettent des décompositions disjointes simples par l'intermédiaire d'une même fonction $h(A)$; soient $A = \{a_1,\ldots,a_s\}$ et $|a_i|$, respectivement $|B|$, le nombre d'occurrences de a_i, respectivement des variables B, d'une écriture minimale de $\varphi(A,B)$. Si

$$[h(A)] \leq \frac{|a_1| + \ldots + |a_s|}{\min_i |a_i|} \qquad (1)$$

alors l'une au moins des écritures minimales de $\varphi(A,B)$ admet une décomposition disjointe simple par l'intermédiaire de $h(A)$, $g[h(A),B]$, et on a les égalités :

$$[h(A)] = \frac{|a_1| + \ldots + |a_s|}{\min_i |a_i|} \qquad (1')$$

$$[g(h,B)] = \min_i |a_i| + |B|, \qquad (2')$$

$$[\varphi(A,B)] = \min_i |a_i| [h(A)] + |B|. \qquad (3')$$

Démonstration.

Soit $f(A,B)$ une écriture minimale de $\varphi(A,B)$:

$\underline{\varphi}(A,B) \leq f(A,B) \leq \overline{\varphi}(A,B)$.

Posons

$$\begin{cases} \underline{\varphi}(A,B) = \underline{g}[h(A),B] \\ \overline{\varphi}(A,B) = \overline{g}[h(A),B] \end{cases}$$

en remarquant que la notation est justifiée puisque $\underline{\varphi}(A,B) \leq \overline{\varphi}(A,B)$ entraîne $\underline{g}(h,B) \leq \overline{g}(h,B)$, et soit u_i une valeur des variables $A-a_i$ telle que :

$$h(a_i,u_i) = \tilde{a}_i ,$$

où $\tilde{a}_i$ désigne exclusivement a ou a' .
On déduit :

$$\underline{g}(\tilde{a}_i,B) \leq f(a_i,u_i,B) \leq \overline{g}(\tilde{a}_i,B)$$

$$\underline{g}[h(A),B] \leq f[h(A),u_i,B] \leq \overline{g}[h(A),B]$$

Posons

$$[\varphi(A,B)] = [f(A,B)] = |a_1| + \ldots + |a_s| + |B| ,$$

où $|a_i|$ est le nombre des occurrences de a_i de f(A,B), $|B|$ le nombre d'occurrences des variables B de f(A,B). Alors, $\forall i \in \{1,\ldots,s\}$:

$$[f(a_i,u_i,B)] \leq |a_i| + |B| \qquad (2)$$

$$[f[\tilde{h}(A),u_i,B]] \leq |a_i| \; [h(A)] + |B| \qquad (3)$$

Si (1) est vérifié, (3) donne pour les i tels que $|a_i| = \min_i |a_i|$:

$$[f[\tilde{h}(A),u_i,B]] \leq |a_1| + \ldots + |a_s| + |B| = [\varphi(A,B)]$$

et $f[\tilde{h}(A),u_i,B] = g[h(A),B]$ est une écriture minimale de $\varphi(A,B)$.

La dernière inégalité ne pouvant être qu'une égalité, (1), (2) et (3) deviennent (1'), (2') et (3').

Cas particulier 1. $[h(A)] = s$

(1) est vérifié et (1') devient :

$$s = \frac{|a_1| + \ldots + |a_s|}{\min_i |a_i|}$$

donc $|a_1| = \ldots = |a_s|$.

On retrouve le théorème 3 de [L1], plus explicitement :

Soit une fonction incomplète $\varphi(A,B)$ dont les bornes admettent des décompositions disjointes simples par l'intermédiaire d'une même fonction monotone et privilégiée $h(A)$. Alors il en est de même pour l'une au moins de ses écritures minimales ; si $A = \{a_1,\ldots,a_s\}$ et si $|a_i|$, respectivement $|B|$, est le nombre d'occurrences de a_i, respectivement des variables B, dans une écriture minimale de $\varphi(A,B)$, on a :

$$|a_1| = |a_2| = \ldots = |a_s|$$

$$[\varphi(A,B)] = |a_i|\,[h(A)] + |B|\,.$$

Cas particulier 2. $\min_i |a_i| = 1$

(1) est vérifié puisqu'il existe une valeur v de B telle que

$$g[h(A),v] = \tilde{h}(A)$$

d'où : $[\tilde{h}(A)] = [g[h(A),v]] \leq [\varphi(A,v)] \leq |a_1| + \ldots + |a_s|$.

(2') devient : $[g(h,B)] = 1 + |B|$

(3') devient : $[\varphi(A,B)] = [h(A)] + |B|$.

On trouve la généralisation suivante du théorème 3 de [B] :

Soit une fonction incomplète $\varphi(A,B)$ dont les bornes admettent des décompositions disjointes simples par l'intermédiaire d'une même fonction $h(A)$. Si $\varphi(A,B)$ est monotone et privilégiée par rapport à une lettre de A, (c'est-à-dire si elle a une écriture minimale $f(A,B)$ où figure une seule occurrence d'une variable de A), alors l'une au moins de ses écritures minimales est de la forme $g[h(A),B]$; en outre :

$$[h(A)] = |a_1| + \ldots + |a_s|$$

$$[g(h,B)] = 1 + |B|$$

$$[\varphi(A,B)] = [h(A)] + |B|$$

si $A = \{a_1,\ldots,a_s\}$ et si $|a_i|$, respectivement $|B|$, est le nombre d'occurrences de a_i, respectivement des variables B, dans $f(A,B)$.

II - FONCTIONS BOOLEENNES PARTICULIERES ADMETTANT UNE DECOMPOSITION DISJOINTE SIMPLE [P3]

Cette étude est un complément au premier chapitre de [P1].

II-1. Fonctions particulières admettant une décomposition disjointe simple.

a. Fonctions décomposables pseudo-impaires ou pseudo-paires.

Une fonction booléenne simple f(X) sera dite pseudo-impaire si elle est égale à sa duale à la complémentation près des variables, c'est-à-dire si :

$$f(X) = f'(\tilde{X})$$

où $\tilde{X}$ est X après complémentation de certains de ses éléments (évidemment $\tilde{X} \neq X$) . f(X) sera dite pseudo-paire si elle reste identique après complémentation de certaines de ses variables (au moins deux si f(X) dépend effectivement de toutes ses variables, ce que nous supposerons toujours par la suite).

Etant donné que les fonctions pseudo-impaires, respectivement pseudo-paires, peuvent être considérées comme des fonctions impaires, respectivement paires, par rapport aux variables telles que $\tilde{x} = x'$ (les autres étant des paramètres), les résultats obtenus vont être très voisins de ceux de [P1].

Théorème 1.

Une condition nécessaire et suffisante pour qu'une fonction décomposable f(A,B) = g[h(A),B] soit pseudo-impaire est :

$$\underline{g(h,B) = g'(h,\widetilde{B})}$$

ou $\underline{g(h,B)}$ et $\underline{h(A)}$ pseudo-impaires.

Démonstration.

$f(A,B) = g[h(A),B]$ pseudo-impaire signifie qu'il existe $\widetilde{A}$ et $\widetilde{B}$ tels que :

$$f(A,B) = f'(\widetilde{A},\widetilde{B})$$

$$g[h(A),B] = g'[h(\widetilde{A}),\widetilde{B}] \qquad (1)$$

Il en résulte les deux cas suivants s'excluant mutuellement :

1. ou $h(A) = h(\widetilde{A})$ (ce qui est trivialement vérifié en prenant $\widetilde{A} = A$) et (1) devient :

   $$g(h,B) = g'(h,\widetilde{B})$$

c'est-à-dire, en particulier, $g(h,B)$ est pseudo-impaire.

2. ou $h(A) = h'(\widetilde{A})$ (c'est-à-dire $h(A)$ pseudo-impaire) et (1) devient :

   $$g(h,B) = g'(h',\widetilde{B})$$

c'est-à-dire $g(h,B)$ pseudo-impaire.

Réciproquement $g(h,B) = g'(h,\widetilde{B})$ entraîne :

$$g[h(A),B] = g'[h(A),\widetilde{B}]$$
$$f(A,B) = f'(A,\widetilde{B})$$

c'est-à-dire $f(A,B)$ pseudo-impaire. $g(h,B)$ et $h(A)$ pseudo-impaires entraînent aussi : $g[h(A),B]$ pseudo-impaire.

Exemple 1.

Le seul type de fonctions de trois variables décomposables et pseudo-impaires est : $xyb + (x' + y')b'$.

* $g(h,b) = hb + h'b'$ vérifie : $g(h,b) = g'(h,b')$

* $h(x,y) = xy$ n'est pas pseudo-impaire.

Exemple 2.

Les types de fonctions de quatre variables décomposables et pseudo-impaires sont :

$$xy + (x + y)(zt' + z't)$$
$$(x + y)z + zt + t(x'y')$$
$$(xy' + x'y)(z + t) + z't$$

et tous les types de la forme : $f(x,y,z) \oplus t$ puisque :

$$f(x,y,z) \oplus t = [f(x,y,z) \oplus t']'.$$

Remarque.

Les fonctions impaires sont des fonctions pseudo-impaires particulières. Nous ne les avons pas énumérées dans les deux exemples précédents.

Théorème 3.

Une condition nécessaire et suffisante pour qu'une fonction décomposable $f(A,B) = g[h(A),B]$ soit pseudo-paire est :

$g(h,B) = g(h,\widetilde{B})$

ou $h(A)$ pseudo-impaire.

Démonstration.

Elle est semblable à la précédente.

b. On peut aussi se demander ce qu'il advient quand on permute des variables de A (le résultat de la permutation sera écrit : $\overline{A}$) et des variables de B (pour obtenir $\overline{B}$) ou, plus généralement, quand, en plus, on complémente certaines variables de A et B (le résultat de l'opération sera écrit : $\overset{\circ\circ}{A}$ et $\overset{\circ\circ}{B}$). Enonçons par exemple le correspondant du théorème 2 :

Théorème 2.

Une condition nécessaire et suffisante pour que $f(A,B) = g[h(A),B]$ soit égale à $f(\overset{\circ\circ}{A},\overset{\circ\circ}{B})$ est :

$$g(h,B) = g(h,\overset{\circ\circ}{B}) \text{ et } h(A) = h(\overset{\circ\circ}{A})$$

ou $$g(h,B) = g(h',\overset{\circ\circ}{B}) \text{ et } h(A) = h'(\overset{\circ\circ}{A}).$$

Exemple.

Considérons les deux fonctions suivantes :

$$\begin{cases} h(x,y) = xy' + x'y = h(y,x) = h(x',y') = h(y',x') \\ \qquad\qquad = h'(x,y') = h'(y',x) = h'(x',y) = h'(y,x') \\ g(h,z,t) = hz + h't + zt = g(h',t,z) \end{cases}$$

et la fonction composée

$$f(x,y,z,t) = g[h(x,y),z,t].$$

Elle vérifie donc :

$$\begin{aligned} f(x,y,z,t) &= f(y,x,z,t) \\ &= f(x',y',z,t) \\ &= f(y',x',z,t) \\ &= f(x,y',t,z) \\ &= f(y',x,t,z) \\ &= f(x',y,t,z) \\ &= f(y,x',t,z). \end{aligned}$$

Remarque.

Les théorèmes précédents sont utiles pour la recherche des permutations et symétries conservant une fonction décomposable et serviront en II-2 pour trouver le nombre des fonctions de quatre variables d'un type décomposable donné.

c. Fonctions décomposables linéaires.

Théorème 4.

Une condition nécessaire et suffisante pour qu'une fonction décomposable f(A,B) = g[h(A),B] soit linéaire est que g(h,B) et h(A) soient linéaires.

Démonstration évidente puisque :

g(h,B) linéaire <=> $g(h,B) = h \oplus B^{\oplus} \oplus a$

h(A) linéaire <=> $h(A) = A^{\oplus} \oplus b$

f(A,B) linéaire <=> $f(A,B) = A^{\oplus} \oplus B^{\oplus} \oplus c$,

où $A^{\oplus}$ et $B^{\oplus}$ sont les condensés disjonctifs de A et B et où a, b et c sont des constantes booléennes.

d. Monômes premiers obligatoires et interdits de première espèce d'une fonction décomposable.

Théorème.

Une condition nécessaire et suffisante pour qu'un monôme premier $m_A n_B$ d'une fonction décomposable f(A,B)=g[h(A),B] soit obligatoire est que les monômes premiers correspondants $\tilde{h}n_B$ et m_A de respectivement g(h,B) et $\tilde{h}(A)$ soient obligatoires.

Démonstration.

La condition est nécessaire. En effet, si $\widetilde{h}\ n_B$ et m_A n'étaient pas monômes premiers obligatoires de respectivement g(h,B) et $\widetilde{h}(A)$, il existerait une base première de g(h,B) et une base première de $\widetilde{h}(A)$ ne les contenant pas, ce qui conduirait à une base de f(A,B) ne contenant pas $m_A\ n_B$: c'est contraire à l'hypothèse $m_A\ n_B$ monôme premier obligatoire du f(A,B).

La condition est suffisante, c'est-à-dire si $\widetilde{hB}$ et $\widetilde{A}$ sont des points de respectivement g(h,B) et $\widetilde{h}(A)$ couverts par les seuls monômes premiers $\widetilde{h}\ n_B$ et m_A, alors le point $\widetilde{A}\ \widetilde{B}$ de f(A,B) est couvert par le seul monôme premier $m_A\ n_B$ de f(A,B). En effet, supposons $\widetilde{A}\ \widetilde{B}$ couvert par un monôme premier $p_A\ q_B$ de f(A,B) ; p_A étant monôme premier de $\widetilde{h}(A)$ et couvrant $\widetilde{A}$, est m_A et $\widetilde{h}\ q_B$, étant monôme premier de g(h,B) et couvrant $\widetilde{h}\ \widetilde{B}$ (puisque q_B couvre $\widetilde{B}$), est $\widetilde{h}\ n_B$.

Théorème 6.

Une condition nécessaire et suffisante pour qu'un monôme premier $m_A\ n_B$ d'une fonction décomposable f(A,B) = g[h(A),B] soit interdit de première espèce est que l'un des monômes premiers correspondants $\widetilde{h}\ n_B$ et m_A de respectivement g(h,B) et h(A) soient interdits de première espèce.

Démonstration.

La condition est nécessaire. En effet, si $m_A\ n_B$ est un monôme premier interdit de première espèce de f(A,B), chacun de ses points appartient à un monôme premier obligatoire de f(A,B). donc d'après le théorème 5, chacun des points de $\widetilde{h}\ n_B$ et m_A appartient à un monôme premier obligatoire respectivement de g(h,B) et $\widetilde{h}(A)$.

La condition est suffisante, c'est-à-dire si $\widetilde{h}\ n_B$ ou m_A est un monôme premier interdit de première espèce de respectivement g(h,B) ou $\widetilde{h}(A)$, alors $m_A\ n_B$ est un monôme premier interdit de première espèce de f(A,B) : ceci résulte de la condition suffisante du théorème 5.

Exemple :

Soit g[h(a,b,c),x,y] = (x+y)(ab+bc+a'c) + x'y

g(h,x,y) = xh + yh + x'y admet xh et x'y pour monômes premiers obligatoires, yh pour monôme premier interdit de première espèce ; h(a,b,c) = ab + bc + a'c admet de même ab et ca' pour monômes premiers obligatoires, bc pour monôme premier interdit de première espèce. Donc xab, xa'c et x'y sont des monômes premiers obligatoires de g[h(a,b,c),x,y], yab, ybc, et ya'c des monômes premiers interdits de première espèce.

II-2. Les fonctions booléennes de quatre variables admettant une décomposition disjointe simple.

Voici la liste des 80 types propres de fonctions booléennes de quatre variables admettant une décomposition disjointe simple ($x \equiv y \equiv z$, par exemple, indique que les fonctions du type considéré sont respectées par l'une quelconque des permutations des variables x, y et z, à des complémentations près ; la colonne des types de fonctions de trois variables générateurs s'explique par le fait que g[h(A),B] est obtenue à partir des fonctions g(h,B) et h(A) à respectivement deux et trois variables ou trois et deux variables).

Représentant du type	Caractéristiques des fonctions du type		Types de fonctions de 3 variables générateurs du type	Nombre de fonctions du type
xyz + t	monotonie,	sur-imparité	3A	64
(x+y+z)t	$x \equiv y \equiv z$	sous-imparité	3A*	64
x+y+z+t	monotonie,	sur-imparité	3A*	16
xyzt	$x \equiv y \equiv z \equiv h$	sous-imparité	3A	16

$(xy+x'y')z+t$	monotonie en z ett,	sur-imparité	3B	96
$(xy'+x'y+z)t$	$x \equiv y$	sous-imparité	$3B^*$	96
$xy'+x'y+z+t$	monotonie en z et t	sur-imparité	$3B^*, 3A^*$	48
$(xy+x'y')zt$	$z \equiv y, z \equiv t$	sous-imparité	3B, 3A	48
$xyz+x'y'z'+t$	monotonie en t,	sur-imparité	3C	32
$(xy'+yz'+zx'+xz'+yx'+zy')t$	$x \equiv y \equiv z$	sous-imparité	$3C^*$	32
$xy'+yz'+zx'+xz'+yx'+zy'+t$	monotonie en t,	sur-imparité	$3C^*$	32
$(xyz+x'y'z')t$	$x \equiv y \equiv z$	sous-imparité	3C	32
$(x+y)z+t$	monotonie	sur-imparité	3D	192
$(xy+z)t$	$x \equiv y$	sous-imparité	$3D^*$	192
$xy+z+t$	monotonie	sur-imparité	$3D^*, 3A^*$	96
$(x+y)zt$	$z \equiv y, z \equiv t$	sous-imparité	3D, 3A	96
$xyz+y'z'+t$	monotonie en x et t,	sur-imparité	3E	192
$(xy'+xz'+yz'+y'z)t$	$y \equiv z$	sous-imparité	$3E^*$	192
$xy'+xz'+yz'+y'z+t$	monotonie en x et t,	sur-imparité	$3E^*$	192
$(xyz+y'z')t$	$y \equiv z$	sous-imparité	3E	192
$xy'z'+x'yz'+x'y'z+t$	monotonie en t,	sur-imparité	3F	64
$(xyz+x'y'+y'z'+z'x')t$	$x \equiv y \equiv z$	sous-imparité	$3F^*$	64

xyz+x'y'+y'z'+z'x'+t	monotonie en t,	sur-imparité	3F*	64
(xy'z'+x'yz'+x'y'z)t	x ≡ y ≡ z	sous-imparité	3F	64
xy+yz+zx+t	monotonie	sur-imparité	3G	64
(xy+yz+zx)t	x ≡ y ≡ z	sous-imparité	3G	64
(xy'+x'y)z+(xy+x'y')z'+t	monotonie en t,	sur-imparité	3H	16
[(xy'+x'y)z+(xy+x'y')z]t	x ≡ y ≡ z	sous-imparité	3H	16
xy+yz+zx'+t	monotonie en y,z et t,	sur-imparité	3I	192
(xy+yz+zx')t	y ≡ z	sous-imparité	3I	192
xyz+(x'+y')z'+t	monotonie en t,	sur-imparité	3J	192
[xyz+(x'+y')z']t	x ≡ y	sous-imparité	3J	192
xy'+x'y+(x+y)(zt+z't')+(x'+y')(zt'+z't)	x ≡ y,z ≡ t	sur-imparité	3C*	24
xy(zt+z't')+x'y'(zt'+z't)		sous-imparité	3C	24
xy'+(x+y)(zt+z't')+y'(zt'+z't)	monotonie en x,	sur-imparité	3E*	96
xy(zt+z't')+y'(zt'+z't)	z ≡ t	sous-imparité	3E	96
xy(zt)+(x'+y')(z'+t')+x'y'	x≡y, z≡t	sur-imparité	3F*	96
(xy'+x'y)z't'+x'y'(z+t)		sous-imparité	3F	96
xy+(x+y)(z+t)	monotonie	sur-imparité	3G	96
xy+(x+y)zt	x≡y, z≡t	sous-imparité	3G	96

xy(z+t)+z't	monotonie en x, y et t	3I	192
(x+y)(z'+t)+zt	x ≡ y	3I	192
xy(z'+t')+(x'+y')zt	x ≡ y, z ≡ t	3J	48
x'y'z't'+(x+y)(z+t)		3J	48
(xy+x'y')(z+t)	monotonie en z et t,	3B,3D	48
xy'+x'y+zt	x ≡ y, z ≡ t	3B*,3D*	48
(xy+x'y')(zt+z't')	x ≡ y, z ≡ t	3B	12
xy'+x'y+zt'+z't		3B*	12
xyzt+x'y'(z'+t')	x ≡ y, z ≡ t	3C	96
xy'+x'y+(x+y)z't'+(x'+y')(z+t)		3C*	96
(x+y)(z+t)	monotonie	3D	48
xy+zt	x ≡ y, z ≡ t	3D*	48
xyzt+z't'	monotonie en x et y	3E	96
(x+y)(z'+t')+zt'+z't	x ≡ y, z ≡ t	3E*	96
(x+y)zt+z't'	monotonie en x et y	3E	96
xy(z'+t')+zt'+z't	x ≡ y, z ≡ t	3E*	96
(xy+x'y')zt+z't'	x ≡ y, z ≡ t	3E	48
(xy'+x'y)(z'+t')+zt'+z't		3E*	48
xyzt+y'(z'+t')	monotonie en x	3E	192
xy'+(x+y)(z't')+y'(z+t)	z ≡ t	3E*	192
xy(z+t)+y'z't'	monotonie en x	3E	192
xy'+(x+y)(z'+t')+y'(zt)	z ≡ t	3E*	192

$(xy'+x'y)(z+t)+x'y'z't'$ $xy(z'+t')+(x'+y')zt+x'y'$	$x \equiv y,\ z \equiv t$	3F 3F*	96 96
$(xy'+x'y)(zt+z't')+x'y(zt'+z't)$ $xy(zt+z't')+(x'+y')(zt'+z't)+x'y'$	$x \equiv y,\ z \equiv t$	3F 3F*	48 48
$xy+(x+y)(zt'+z't)$	monotonie en x et y, $x \equiv y, z \equiv t$	3G	48
$(x+y)z+zt+t(x'y')$	monotonie en z et t, $x \equiv y$	3I	192
$(xy'+x'y)z+zt+t(xy+x'y')$	monotonie en z et t $x \equiv y,\ z \equiv t$, imparité	3I	48
$(xy'+x'y)(z+t)+z't$	monotonie en t $\quad x \equiv y$	3I	96
$xyz \oplus t$	$x \equiv y \equiv z$	3A, 3A*, 3J	64
$(xy'+x'y+z) \oplus t$	$x \equiv y$	3B, 3B*, 3J	96
$(xyz+x'y'z') \oplus t$	$x \equiv y \equiv z$ imparité	3C, 3C*	32
$(x+y)z \oplus t$	$x \equiv y$	3D, 3D*	192
$(xyz+y'z') \oplus t$	$y \equiv z$	3E, 3E*	192
$(xyz+x'y'+y'z'+z'x') \oplus t$	$x \equiv y \equiv z$	3F, 3F*	64
$(xy+yz+zx) \oplus t$	$x \equiv y \equiv z$	3G	32
$x \oplus y \oplus z \oplus t$	$x \equiv y \equiv z \equiv t$	3H	2
$(xy+yz+zx') \oplus t$		3I	96
$xy \oplus z \oplus t$	$x \equiv y,\ z \equiv t$	3J, 3H	48

Sur 64 594 fonctions dépendant effectivement de quatre variables, il y en a donc 7 354 qui admettent une décomposition disjointe simple non triviale. La borne supérieure donnée dans [1,2] de la proportion des fonctions de quatre variables admettant une décomposition disjointe simple parmi les fonctions de quatre variables

$$p(4) = \frac{40}{252} \simeq 16\ \%$$

est donc à comparer avec :

$$\frac{7\ 354}{64\ 594} \simeq 11\ \% .$$

III - NOUVEL ALGORITHME POUR LA DECOMPOSITION SIMPLE DES FONCTIONS BOOLEENNES [P,D]

[P1,2] donne une condition nécessaire et suffisante pour qu'une fonction booléenne simple $f(A,B,Z)$ admette la décomposition simple $g[h(A,Z),B,Z]$. Nous la réénonçons en employant, à la suite de [D], la notation :

$$\frac{\Delta k(x,X)}{\Delta x}$$

et l'appellation dérivée par rapport à x de $k(x,X)$ pour le coefficient de la variable x dans l'écriture galoisienne de $k(x,X)$ ou coefficient galoisien de x (il est égal à $\frac{k(0,X) \oplus k(1,X)}{0 \oplus 1} = k(0,X) \oplus k(1,X)$) :

Théorème 1.

Une condition nécessaire et suffisante pour que la fonction booléenne simple $f(A,B,Z)$ admette la décomposition simple $g[h(A,Z),B,Z]$ est que, pour toute variable a_i de $A(i=1,\ldots,s)$, $\frac{\Delta f(A,B,Z)}{\Delta a_i}$ soit de la forme $a(B,Z)h_i(A-a_i,Z)$, où $a(B,Z)$ et $h_i(A-a_i,Z)$ sont des fonctions de variables de, respectivement, $B \cup Z$ et $(A-a_i) \cup Z$.

Nous allons donner une nouvelle condition nécessaire et suffisante de décomposabilité permettant en particulier de simplifier l'algorithme de recherche des décompositions disjointes simples de [P1,2] fondé sur le théorème 1.

Dans toute la suite de l'exposé, ne considérant que des fonctions et des décompositions simples, nous omettrons de le signaler.

III-1. Une condition simple de décomposabilité.

a. Unicité de la décomposition additive d'une fonction complète

Lemme 1

Une condition nécessaire et suffisante pour que

$$h(A,Z) + k(B,Z) = h_1(A,Z) + k_1(B,Z) \qquad (1)$$

entraîne

$$\begin{cases} h_1(A,Z) = h(A,Z) \\ k_1(B,Z) = k(B,Z) \end{cases} \qquad (2)$$

est que la fonction $h(A,Z) + k(B,Z)$ n'ait pas de monôme ne contenant que des lettres de Z (ou égal à 1).

Démonstration.

La condition est nécessaire : en effet, supposons que m_Z soit un monôme ou une somme de monômes de $h(A,Z) + k(B,Z)$ ne contenant que des lettres de Z et construisons à partir de $h(A,Z)$ les deux fonctions distinctes suivantes :

$$h_1(A,Z) = h(A,Z) + m_Z$$

$$h_2(A,Z) = h(A,Z)\,(m_Z)'.$$

Alors :

$$h(A,Z) + k(B,Z) = h_1(A,Z) + k(B,Z) = h_2(A,Z) + [m_Z + k(B,Z)]$$

et (1) n'entraîne donc pas (2) puisque

$$h_1(A,Z) \neq h_2(A,Z).$$

Réciproquement, (1) entraîne, pour toute configuration $\tilde{Z}$ des variables Z :

$$h(A,\tilde{Z}) + k(B,\tilde{Z}) = h_1(A,\tilde{Z}) + k_1(B,\tilde{Z}). \qquad (3)$$

Puisque $h(A,Z) + k(B,Z)$ n'admet pas de monôme premier de la forme m_Z, il en résulte par l'absurde que $h(A,\tilde{Z}) \neq 1$; il existe donc une configuration $\tilde{A}$ des variables A telle que $h(\tilde{A},\tilde{Z}) = 0$ et (3) devient :

$$k(B,\tilde{Z}) = h_1(\tilde{A},\tilde{Z}) + k_1(B,\tilde{Z}).$$

$k(B,\tilde{Z})$ étant pour la même raison différent de 1, $h_1(\tilde{A},\tilde{Z}) = 0$, ce qui entraîne :

$$k(B,\tilde{Z}) = k_1(B,\tilde{Z}). \qquad (4)$$

On démontrerait semblablement :

$$h(A,\tilde{Z}) = h_1(A,\tilde{Z}). \qquad (5)$$

(4) et (5) vrais pour tout $\tilde{Z}$ entraînent (2).

Ce lemme a évidemment une forme duale qu'on peut exprimer de la façon suivante :

Lemme 1'.

Une condition nécessaire et suffisante pour que

$$h(A,Z) \cdot k(B,Z) = h_1(A,Z) \cdot k_1(B,Z)$$

entraîne

$$\begin{cases} h_1(A,Z) = h(A,Z) \\ k_1(B,Z) = k(B,Z) \end{cases}$$

est que $h^*(A,Z) + k^*(B,Z)$ n'ait pas de monôme (premier) ne contenant que des lettres de Z.

A ces lemmes 1 et 1', relatifs aux décompositions simples additives ou multiplicatives, correspond le lemme suivant relatif aux décompositions simples disjonctives :

lemme 1".

Une condition nécessaire et suffisante pour que

$$h(A,Z) \oplus k(B,Z) = h_1(A,Z) \oplus k_1(B,Z)$$

entraîne :

$$\begin{cases} h_1(A,Z) = \tilde{h}(A,Z) \\ k_1(B,Z) = \tilde{k}(B,Z) \end{cases}$$

est que l'écriture galoisienne de $h(A,Z) \oplus k(B,Z)$ n'ait pas de monôme ne contenant que des lettres de Z.

Sa démonstration est analogue à celle du lemme 1.

Remarque 1.

Que la fonction $h(A,Z) + k(B,Z)$ n'ait pas de monôme ne contenant que des lettres de Z est équivalent à : les fonctions $h(A,Z)$ et $k(B,Z)$ séparément n'ont pas de monôme ne contenant que des lettres de Z. Cela résulte en particulier de ce que tout monôme premier de $h(A,Z) + k(B,Z)$ peut être obtenu à partir des monômes premiers de $h(A,Z)$ et de $k(B,Z)$ à l'aide de consensus par rapport aux seules lettres Z (ne supprimant donc que des lettres Z).

Remarque 2.

Le cas particulier où Z est vide (cas des décompositions additives, multiplicatives ou disjonctives disjointes) est dans [C].

Remarque 3.

Si Card Z = 1, les conditions des lemmes 1 et 1' deviennent respectivement : h(A,Z) + k(B,Z) n'est pas égal à 1 ou n'admet pas de décomposition additive disjointe $\widetilde{z} + \ell(A,B)$, h(A,Z) k(B,Z) n'est pas nul ou n'admet pas de décomposition multiplicative disjointe $\widetilde{z}\ell(A,B)$.
Puisque pratiquement les décompositions disjointes sont recherchées avant les décompositions non disjointes, les conditions des lemmes 1 et 1', quand Card Z = 1, sont automatiquement vérifiées.

Remarque 4.

Le lemme 1 peut être considéré comme une conséquence du résultat suivant de [P1] :

Lemme 2.

Soit f(X) = h(A,Z) + k(B,Z) une fonction booléenne admettant une décomposition additive suivant la partition (A,B,Z). Appelons $m_{A(Z)}$, $m_{B(Z)}$ et m_Z tous les monômes premiers de f(X) contenant respectivement au moins une variable de A, au moins une variable de B et seulement des variables de Z. Alors :

$$\begin{cases} m_{A(Z)} \cdot (m_Z)' \leq h(A,Z) \leq m_{A(Z)} + m_Z \\ m_{B(Z)} \cdot (m_Z)' \leq k(B,Z) \leq m_{B(Z)} + m_Z \end{cases}.$$

b. Fonctions incomplètes décomposables additivement.

On peut encore poser le problème plus général suivant : étant données la fonction incomplète $\varphi(X)$ et la partition (A,B,Z) des variables X, quelles sont les décompositions additives suivant (A,B,Z) de $\varphi(X)$? Il est facile d'y répondre.

De même qu'on a ramené, en I-2, le problème de la détermination d'une écriture minimale d'une fonction booléenne à celui de la détermination

d'une écriture croissante minimale de la fonction incomplète croissante attachée, nous pouvons ramener la recherche d'une décomposition additive non disjointe suivant une partition donnée à celle d'une décomposition additive disjointe grâce à la propriété suivante de [P1] : une condition nécessaire et suffisante pour que la fonction incomplète $\varphi(X)$ admette la décomposition additive $h(A,Z) + k(B,Z)$, est que la fonction incomplète

$$\gamma(A,B,Z,U) = \begin{cases} \varphi(X) & \text{si } U = Z \\ \not{0} & \text{si } U \neq Z \end{cases}$$

où $\not{0}$ désigne indifféremment 0 ou 1 (c'est la valeur caractéristique d'une fonction incomplète), admette la décomposition additive disjointe $h(A,Z) + k(B,U)$. Il est alors facile de répondre à la question posée puisque [P1] (p.I-22 et suivantes) donne aussi le moyen de déterminer les décompositions additives disjointes suivant une partition donnée d'une fonction incomplète. L'exemple qui suit montre comment procéder.

Exemple :

Cherchons les fonctions additivement décomposables suivant la partition (a,b,z) qui sont compatibles avec la fonction incomplète suivante :

$$\begin{cases} \underline{\varphi}(a,b,z) = ab + az + bz \\ \overline{\varphi}(a,b,z) = a + b + z \end{cases}$$

(les décompositions additives disjointes sont données dans [P1]).

1. La fonction $\gamma(a,b,z,u)$ obtenue à partir de $\varphi(a,b,z)$ par dédoublement de z en u :

$$\begin{cases} \underline{\gamma}(a,b,z,u) = (ab + az + bz)(zu + z'u') = azu + bzu + abz'u' \\ \overline{\gamma}(a,bz,u) = a + b + z + z'u + zu' = a + b + z + u \end{cases}$$

a pour décompositions minimales compatibles suivant (az,bu)[P1] p. I-24 :

$$maz + m'u + pz + p'bu + qaz' + q'bu'$$

(m,p,q sont des paramètres booléens), soit, tout calcul fait :

$$a + z,\ b + u,\ z + bu',\ u + az',\ a + bu,\ b + az.$$

Les décompositions minimales compatibles avec $\varphi(a,b,z)$ sont donc :
$a + bz$ et $b + az$.

2. La décomposition maximale compatible avec $\varphi(a,b,z)$ est :
$a + b + z$.

3. Les fonctions décomposables sont données par :

$$a + bz + \lambda(a+z) + \mu(b+z) \quad \text{et} \quad b + az + \lambda(a+z) + \mu(b+z),$$

où λ et μ sont des paramètres booléens. Ne considérons que la première de ces deux dernières expressions, a et b jouant un rôle symétrique ; elle s'écrit :

$$a+bz+(\lambda_1\lambda_2\lambda_3\lambda_4)\begin{pmatrix} a'z' \\ a'z \\ az' \\ az \end{pmatrix}(a+z) + (\mu_1\mu_2\mu_3\mu_4)\begin{pmatrix} b'z' \\ b'z \\ bz' \\ bz \end{pmatrix}(b+z)$$

$$= a + bz + (\lambda_2 + \mu_2)z + \mu_3 b$$

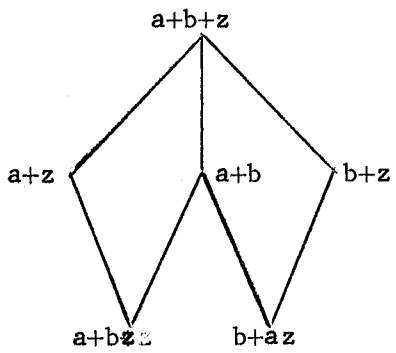
et nous obtenons pour les diverses valeurs de λ_2, μ_2 et μ_3 :

$$a + bz,\ a + b,\ a + z,\ a + b + z.$$

Finalement, les fonctions additivement décomposables suivant (a, b, z) compatibles avec $\varphi(a,b,z)$ sont :

$$a + bz,\ b + az,\ a + b,\ a + z,\ b + z,\ a + b + z$$

et leur sup demi-treillis est :



Déterminons maintenant directement (sans passer par γ) les fonctions décomposables additivement suivant (A,B,Z) et inférieures ou égales à une fonction donnée, le cas des fonctions décomposables supérieures ou égales à une fonction donnée étant plus compliqué.

Lemme 3.

Les fonctions $h(A,Z) + k(B,Z)$ décomposables additivement suivant (A,B,Z) et inférieures ou égales à :

$$\overline{\varphi}(A,B,Z) = \overline{m}_{A(Z)} + \overline{m}_{B(Z)} + \overline{m}_{Z} + \overline{m}_{AB(Z)}$$

où $\overline{m}_{A(Z)}$ par exemple désigne l'ensemble des monômes premiers de $\overline{\varphi}(A,B,Z)$ contenant au moins une variable de A et aucune variable de B, sont inférieures ou égales à :

$$\overline{m}_{A(Z)} + \overline{m}_{B(Z)} + \overline{m}_{Z} \qquad (1)$$

et $h(A,Z)$ et $k(B,Z)$ vérifient :

$$\begin{cases} h(A,Z) \leq \overline{m}_{A(Z)} + \overline{m}_{Z} & (2) \\ k(B,Z) \leq \overline{m}_{B(Z)} + \overline{m}_{Z} & (3) \end{cases}$$

Démonstration :

(2) résulte de ce que tout monôme premier de $h(A,Z)$ ne contient que des lettres de A et de Z et est inférieur ou égal à au moins un monôme

premier de $h(A,Z) + k(B,Z)$.

(3) est vrai puisque (2) l'est.

Enfin (1) résulte de (2) et (3) et de sa décomposabilité suivant (A,B,Z). Remarquons que (2) et (3) sont, à l'encontre de (1), obligatoirement des bases complètes.

c. Condition de décomposabilité d'une fonction complète.

Théorème 2.

Une condition suffisante pour que la fonction complète $f(A,B,Z)$ admette la décomposition simple $g[h(A,Z),B,Z]$ est que le coefficient galoisien $\dfrac{\Delta f(A,B,Z)}{\Delta a_i}$ de toute variable a_i de $A(i = 1,\ldots,s)$ admette une décomposition multiplicative suivant (A,B,Z) et qu'il existe une chaîne des variables de A telle que pour tout couple a_i et a_j $(1 \leq i < j \leq s)$ de variables consécutives $\dfrac{\Delta^2 f(A,B,Z)}{\Delta a_i \, \Delta a_j}^{*}$, fonction duale de $\dfrac{\Delta^2 f(A,B,Z)}{\Delta a_i \, \Delta a_j}$, n'ait pas de monôme ne contenant que des lettres de Z.

Démonstration.

Montrons que :

$$\frac{\Delta f(A,B,Z)}{\Delta a_i} = \alpha_i(B,Z)\, h_i(A-a_i,Z), \qquad i = 1,\ldots,s,$$

entraîne :

$$\frac{\Delta f(A,B,Z)}{\Delta a_i} = \alpha(B,Z)\ h_i(A-a_i,Z), \qquad i = 1,\ldots,s$$

si les conditions du théorème 2 sont vérifiées, donc leur suffisance pour l'existence de g[h(A,Z),B,Z] en vertu du théorème 1.

En effet, pour tout couple a_i et a_j de variables consécutives de la chaîne , on a :

$$\frac{\Delta^2 f(A,B,Z)}{\Delta a_i \ \Delta a_j} = \frac{\Delta^2 f(A,B,Z)}{\Delta a_j \ \Delta a_i} ,$$

d'où :

$$\alpha_i(B,Z)\ \frac{\Delta h_i(A-a_i,Z)}{\Delta a_j} = \alpha_j(B,Z)\ \frac{\Delta h_j(Y-a_j,Z)}{\Delta a_i}$$

et, d'après le lemme 1, puisque $\dfrac{\Delta^2 f(A,B,Z)}{\Delta a_i \ \Delta a_j}$ n'a pas de monôme ne contenant que des lettres de Z :

$$\alpha_i(B,Z) = \alpha_j(B,Z).$$

Donc

$$\alpha_1(B,Z) = \alpha_2(B,Z) = \ldots = \alpha_s(B,Z)$$

Les théorèmes 1 et 2 permettent évidemment d'énoncer, si ϕz désigne z ou rien (respectivement d'une décomposition non disjointe avec une seule variable non disjointe et d'une décomposition disjointe) et en tenant compte de la remarque 3 :

Théorème 3.

Une condition suffisante pour que la fonction $f(A,B,\Phi z)$ admette la décomposition simple $g[h(A,\Phi z),B,\Phi z]$ est que le coefficient galoisien de toute variable a_i de A ($i = 1,\ldots,s$) admette une décomposition multiplicative suivant la partition $(A,B,\Phi z)$ et qu'il existe une chaîne des variables de A telles que pour deux variables consécutives a_i et a_j $\dfrac{\Delta^2 f}{\Delta a_i \, \Delta a_j}$ soit non nul.

A l'aide des théorèmes 2 ou 3, la recherche des décompositions devient facile ; si tous les coefficients galoisiens des variables de A admettent une décomposition multiplicative suivant (A,B,Z) mais s'il n'existe pas une chaîne telle que pour deux variables consécutives a_i et a_j, $\dfrac{\Delta^2 f(A,B,Z)}{\Delta a_i \, \Delta a_j}$ * n'ait pas de monômes ne contenant que des lettres de Z, il faut cependant de nouveau utiliser le théorème 1.

Donnons maintenant en détail un algorithme fournissant les seules décompositions disjointes, en utilisant le fait qu'elles sont ordonnées en treillis, d'une façon que l'on va rappeler :

III-2. Algorithme donnant les classes des décompositions disjointes simples totales d'une fonction complète.

a. Rappels et définitions.

On démontre (voir [C]) que, si $\{(A_p,B_p)\ ;\ 1 \leq p \leq P \leq n\}$ est l'ensemble des partitions (A_p,B_p) des décompositions disjointes maximales (c'est-à-dire dont les A, appelés A_p, sont maximaux) non triviales

($1 < \text{Card } A_p < n$) ou triviales ($\text{Card } A_p = 1$), alors ou $\{A_p ; 1 \leq p \leq P\}$ est une partition de X et

$$f(X) = g[f_1(A_1), f_2(A_2), \ldots, f_P(A_P)], \qquad (1)$$

ou $\{B_p ; 1 \leq p \leq P\}$ est une partition de X et

$$f(X) = f_1(B_1) \circ f_2(B_2) \circ \ldots \circ f_P(B_P) \qquad (2)$$

($\circ$ désigne +,. ou $\oplus$). L'expression unique, à la complémentation près des fonctions $f_p(A_p)$ ou $f_p(B_p)$, de f(X), qu'elle soit (1) ou (2), sera appelée <u>la décomposition disjointe (simple) totale</u> [K] de f(X) (elle est triviale si, quel que soit p, $\text{Card } A_p = 1$), et les sous-ensembles $A_1,\ldots,A_P$ de X dans le cas d'une décomposition de la forme (1) ou $B_1,\ldots,B_p$ dans le cas d'une décomposition de la forme (2), seront dits les <u>classes</u> de la décomposition disjointe totale de f(X).

[C] montre encore que, étant donnée une partition (A,B) d'une décomposition disjointe de f(X), l'un au moins des sous-ensembles A et B de X est une classe de la décomposition disjointe totale de f(X) ou de celle d'une fonction $f_p(X_p)$ ($1 \leq p \leq P$) ou de celle d'une fonction $f_{pq}(X_{pq})$ associée à $f_p(X_p)$... et d'une seule. L'inverse est évidemment exact.

Dans la suite de l'exposé, on appellera par extension l'ensemble des décompositions disjointes totales de f(X), $f_p(X_p)$ ($1 \leq p \leq P$), $f_{pq}(X_{pq})$,... <u>les décompositions disjointes totales</u> de f(X), et l'ensemble de leurs classes <u>les classes des décompositions disjointes totales</u> de f(X).

b. <u>Algorithme</u>.

Considérons, pour toute variable x de X

$$\frac{\Delta f(X)}{\Delta x} = \prod_{j=1}^{J} \alpha_{x,j}(X_{x,j}) \qquad (I)$$

où $\prod_{j=1}^{J} \alpha_{x,j}(X_{x,j})$ est la décomposition disjointe totale multiplicative de $\frac{\Delta f(X)}{\Delta x}$,

$$P_x = (X_{x,1}, \ldots, X_{x,J}) \quad , \quad x \in X \qquad (II)$$

où les P_x sont des partitions de X puisqu'on adoptera la convention suivante : si une variable de X ne figure pas dans $\frac{\Delta f(X)}{\Delta x}$, elle formera à elle seule une classe de P_x.

D'après le théorème 1, le système (I) des coefficients galoisiens dans f(X) de chaque variable x de X décomposés disjonctivement et multiplicativement est nécessaire et suffisant à la recherche des décompositions disjointes totales de f(X). D'après le théorème 3, la donnée du système (II) de partitions de X est ordinairement suffisante pour trouver les classes des décompositions disjointes totales de f(X) ; le passage de (II) et d'au besoin (III) à l'ensemble de ces classes de décomposition est l'objet de ce qui suit.

Appelons la classe de la partition P_x contenant son indice x, la classe spéciale de P_x et définissons dans l'ensemble des partitions de X à indice contenu dans une seule classe l'opération interne faisant correspondre à deux partitions P_L et P_M d'indices L et M la partition réunion des partitions P_L, P_M, $L \cup M$ et d'indice $L \cup M$:

$$P_{L \cup M} = \cup(P_L, P_M, L \cup M).$$

Alors on déduit immédiatement du théorème 3 :

Lemme 1.

La plus petite classe (par rapport à l'inclusion ensembliste) de décomposition disjointe totale contenant deux lettres u et v de X est la classe spéciale de la première partition constructible à partir de P_u et de P_v à l'aide de l'opération interne qui vient d'être définie, et dont l'indice est identique à la classe spéciale, si la classe de P_u contenant v et la classe de P_v contenant u ont au moins un élément commun ou ont respectivement un élément x_i et un élément x_j tels que $\frac{\Delta^2 f}{\Delta x_i \, \Delta x_j}$ soit non nul.

Le lemme suivant se déduit des rappels :

Lemme 2.

Pour toute classe de décomposition A qui n'est pas réduite à une variable et dont la fonction h(A) correspondante (y compris f(X)) n'admet pas de décomposition disjointe totale de la forme (2), il existe toujours deux lettres u et v de X telles que la plus petite classe de décomposition contenant u et v soit identique à A. Inversement, toute plus petite classe de décomposition contenant deux lettres u et v n'appartenant pas à deux classes distinctes d'une décomposition disjointe totale de la forme (2) avec $P > 2$, est une classe de décomposition disjointe totale.

Les lemmes 1 et 2 conduisent respectivement à l'algorithme 1 de recherche de la plus petite classe de décomposition disjointe totale contenant deux lettres u et v de X et à l'algorithme 2 donnant les classes des décompositions disjointes totales d'une fonction :

Algorithme 1.

Etant données les partitions P_u et P_v, on initialise le processus suivant en posant $P_W := P_u$, $P_x := P_v$:

1. A partir des partitions P_W et P_x, on calcule

$P_{W \cup x} = \cup(P_W, P_x, W \cup x)$ de classe spéciale $Y_{W \cup x}$.

2. On effectue le test :

- si Card $Y_{W \cup x}$ = n, la plus petite classe de décomposition contenant u et v est X ;
- sinon et si Card $Y_{W \cup x}$ > Card $(W \cup x)$, on va en 1 avec

$$P_W := P_{W \cup x}$$

$$P_x := P_w \text{ où } w \in Y_{W \cup x} - W \cup x$$

- sinon (Card $Y_{W \cup x}$ = Card$(W \cup x) < n$), la plus petite classe de décomposition contenant u et v est $W \cup x$, si la classe de P_u contenant v et la classe de P_v contenant u ont au moins un élément commun ou ont respectivement des éléments x_i et x_j tels que $\dfrac{\Delta^2 f}{\Delta x_i \, \Delta x_j}$ soit non nul ;

  sinon, on utilise le théorème 3 pour vérifier que $W \cup x$ est effectivement une classe de décomposition ;
- sinon, on ajoute un lettre (de toutes les façons possibles) à $W \cup x$ et on itère le calcul précédent.

Algorithme 2.

En appliquant l'algorithme 1 à tout couple de variables de X, on obtient un ensemble $\mathcal{H}$ de sous-ensembles de X. L'ensemble des classes des décompositions est X et les A_p ou B_p obtenus par le processus suivant : W := X.

1. Si W est un élément de $\mathcal{A}$, les éléments de $\mathcal{A}$ qui précèdent immédiatement W sont des A_p ;

2. Si W n'est pas un élément de $\mathcal{A}$, les complémentaires par rapport à W des éléments de $\mathcal{A}$ qui sont les immédiats prédécesseurs de W sont des B_p ;

3. On repasse en 1 avec W := A_p, B_p, ...

Exemple :

Trouvons les décompositions disjointes simples totales de :

$$f(u,v,w,x,y,z) = u'v'wx'y'z' + u'v'w\,xyz' + u'vw'z' + u'vwx'y'z + u'vwxyz + uv'w'x'y'z' + uv'w'xyz' + uvw'x'y'z + uvwz' + vx'yz' + vxy'z' + yvw'xyz.$$

Les décompositions disjointes totales additives des duaux des coefficients galoisiens des variables sont :

$$\left(\frac{\Delta f(u,v,w,x,y,z)}{\Delta u}\right)^* = [vz'] + [x'y + xy']$$

$$\left(\frac{\Delta f(u,v,w,x,y,z)}{\Delta v}\right)^* = [u'w'z' + uwz' + x'yz' + xy'z']$$

$$\left(\frac{\Delta f(u,v,w,x,y,z)}{\Delta w}\right)^* = [vz'] + [x'y + xy']$$

$$\left(\frac{\Delta f(u,v,w,x,y,z)}{\Delta x}\right)^* = [u'w' + uw] + [vz']$$

$$\left(\frac{\Delta f(u,v,w,x,y,z)}{\Delta y}\right)^{*} = [u'w' + uw] + [vz']$$

$$\left(\frac{\Delta f(u,v,w,x,y,z)}{\Delta z}\right)^{*} = [u'vw' + uvw + wx'y + vxy'].$$

et les partitions correspondantes :

$P_u = (vz, xy)$

$P_v = (uwxyz)$

$P_w = (vz, xy)$

$P_x = (uw, vz)$

$P_y = (uw, vz)$

$P_z = (uvwxy)$

L'algorithme 1 donne :

$P_{uv} = P_{vw} = p_{vx} = P_{vy} = P_{vz} = P_{uz} = P_{wz} = P_{xz} = P_{yz} = (uvwxyz)$

$P_{uw} = (uw, vz, xy)$

$P_{ux} = (uwxy, vz) = P_{uwx} = P_{uwxy}$

$P_{uy} = (uwxy, vz) = P_{uwy} = P_{uwxy}$

$P_{wx} = (uwxy, vz) = P_{uwx} = P_{uwxy}$

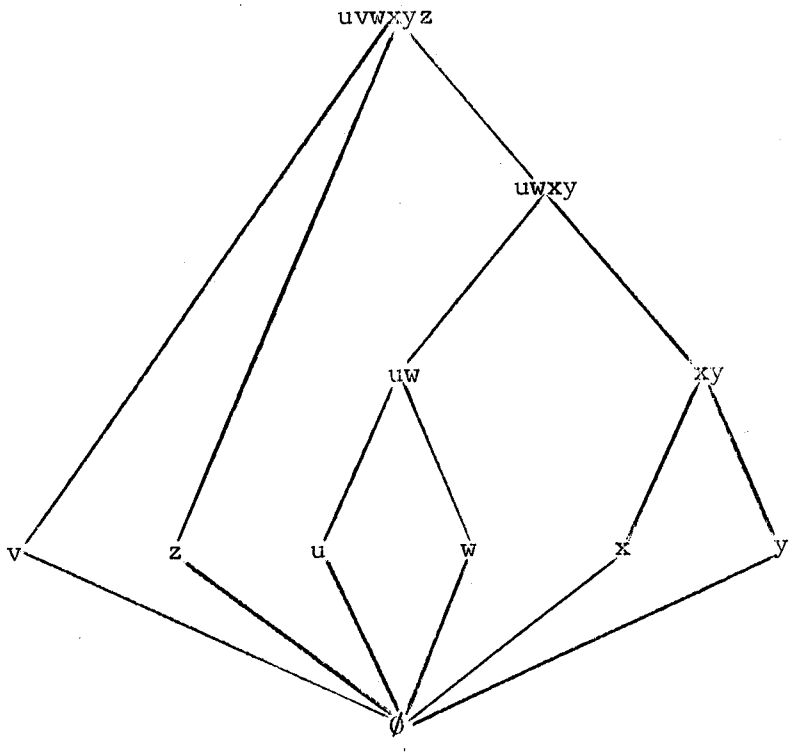
$P_{wy} = (uwxy, vz) = P_{uwy} = P_{uwxy}$

$P_{xy} = (xy, uw, vz)$

L'ensemble des plus petites classes de décomposition disjointe totale contenant deux lettres est donc :

uvwxyz, uw, uwxy, xy.

L'algorithme 2 donne alors comme treillis des classes des décompositions de f(u,v,w,x,y,z) :



L'écriture totalement décomposée de f(u,v,w,x,y,z) est donc :

$$f(u,v,w,x,y,z) = g[h(u,w,x,y),v,z]$$

où
$$\begin{cases} g(h,v,\varnothing) = h'vz' + hv'z' + hvz \\ h(u,w,x,y) = (u \oplus w)\,(x \oplus y \oplus 1). \end{cases}$$

CHAPITRE 2

DECOMPOSITIONS

D'UNE FONCTION DE PLUSIEURS VARIABLES

A propos des décompositions d'une fonction F(X) de plusieurs variables X, c'est-à-dire des expressions de F(X) de la forme G[H(A,Z),B,Z], où (A,B,Z) est une partition de X, deux questions se posent principalement:

1. Comment trouver les décompositions ? Dans le cas où les variables ont un nombre fini de valeurs et étant donné une partition (A,B,Z) des variables, un tableau permet de savoir si F(X) admet une décomposition non triviale suivant cette partition ; dans le cas des fonctions booléennes, rappelons que l'écriture galoisienne, grâce à sa linéarité par rapport à chacune des variables permet de trouver les décompositions simples d'une fonction, sans la donnée préalable de leur partitions (1-III).

2. Comment les décompositions d'une fonction s'agencent-elles les unes par rapport aux autres ? Ou d'un point de vue pratique, quelles décompositions est-il préférable d'utiliser pour la synthèse d'une fonction ? En 1-III-2a, nous avons rappelé les résultats remarquables obtenus à ce sujet en algèbre de Boole ; nous nous intéressons ici à leur généralisation à des fonctions arbitraires.

Remarquons qu'une réponse à la deuxième question aide la résolution de la première, la recherche des décompositions d'une fonction (cette remarque a été utilisée pour l'algorithme de 1-III).

I - PRELIMINAIRES

I-1. Hypothèses

Etant donné un ensemble produit arbitraire $E_1 \times E_2 \times \ldots \times E_n$, nous considérons des fonctions partout définies sur cet ensemble et complètes, c'est-à-dire des fonctions ayant une valeur ou un jeu de valeurs précisées pour tous les n-uplets de $E_1 \times E_2 \times \ldots \times E_n$.

Remarquons qu'il est toujours possible d'associer à une fonction non partout définie ou à une fonction incomplète une fonction partout définie et complète ; en particulier, un choix judicieux des valeurs non spécifiées d'une fonction innomplète peut faire apparaître des décompositions intéressantes.

Les fonctions considérées par la suite sont aussi supposées surjectives ; autrement dit, les données d'une fonction et de son espace objet définissent son espace image.

I-2. Notations

Les lettres minuscules, excepté k, (par exemple x) désigneront des variables simples dont le domaine de définition est arbitraire ou des fonctions simples et complètes de telles variables, les lettres capitales (par exemple X) désigneront des variables générales (ensemble fini et ordonné de variables simples), ou des fonctions générales et complètes. L'ensemble des valeurs de X (ou x) est noté dom X (ou dom x) et leur nombre est noté Card dom X (ou Card dom x). Des valeurs particulières de X (respectivement x) sont notées X_1, X_2, ..., ou X_i, X_j,... (respectivement

$x_1, x_2, \ldots, x_i, x_j, \ldots)$.

Il nous arrivera d'écrire F pour F(X).

I-3. Représentation tabulaire et décomposition.

a. A une fonction F(A,B,Z), on peut faire correspondre un tableau $\mathcal{G}^{F}_{BZ,AZ}$ (symboliquement dans le cas où les variables prennent une infinité de valeurs) dont chacune des lignes correspond à une valeur de {B,Z}, dont chacune des colonnes correspond à une valeur de {A,Z} et dont les éléments sont les valeurs correspondantes de F(A,B,Z) si elles existent, sinon (cas des fonctions non définies ou cas d'incompatibilité des arguments) des blancs.

b. Une décomposition d'une fonction F(A,B,Z) est un couple ordonné de deux fonctions G et H telles que :

$$F(A,B,Z) = G[H(A,Z),B,Z]$$

pour toutes les valeurs des variables A,B et Z. On dira aussi que G[H(A,Z),B,Z] est une décomposition de F(A,B,Z) suivant le triplet ordonné (A,B,Z) partition des variables de F.

Si l'ensemble Z est vide, la décomposition est dite disjointe.

Remarquons que la fonction G(H,B,Z) n'est pas obligatoirement toujours définie, contrairement à F, mais c'est sans importance pour la suite. La fonction H(A,Z) sera appelée fonction composante de la décomposition G[H(A,Z),B,Z].

c. Relativement au tableau représentant la fonction F(A,B) et au tableau représentant sa décomposition disjointe G[H(A),B] (si elle existe), on peut énoncer : tout vecteur colonne $G(H_m,B)$ de $\mathscr{C}^{G}_{B,H}$ est un vecteur colonne de $\mathscr{C}^{F}_{B,A}$ et, inversement, tout vecteur colonne $F(A_i,B)$ de $\mathscr{C}^{F}_{B,A}$ est un vecteur colonne de $\mathscr{C}^{G}_{B,H}$.

Démonstration.

La surjectivité de H entraîne que, quelle que soit sa valeur H_m, il existe au moins une valeur A_i de A telle que

$$H(A_i) = H_m$$

d'où

$$G(H_m,B) = G[H(A_i),B] = F(A_i,B).$$

Inversement

$$F(A_i,B) = G[H(A_i),B] = G(H_m,B]$$

si H_m est la valeur de $H(A_i)$.

Remarque 1.

Card dom H est supérieur ou égal au cardinal des colonnes différentes de $\mathscr{C}^{F}_{B,A}$ et inférieur ou égal à Card dom A.

Remarque 2.

La donnée de la décomposition non disjointe G[H(A,Z),B,Z] et celle de l'ensemble $\{G[H(A,k),B,k] \mid k \in \text{dom } Z\}$ de décompositions disjointes sont équivalentes. Des résultats relatifs aux décompositions disjointes peuvent

donc en fait être aussi généraux que ceux où interviennent des décompositions non disjointes.

I-4. Donnons une généralisation aux décompositions non disjointes, aux variables quelconques et aux fonctions générales d'un lemme de [KA].

Lemme 1.

Une condition nécessaire et suffisante d'existence de la décomposition

(1) $F(A,B,Z) = G[H(A,Z),B,Z]$

est, pour toutes les valeurs A_i et A_j de A et pour toute valeur k de Z :

(2) $H(A_i,k) = H(A_j,k) \Rightarrow F(A_i,B,k) = F(A_j,B,k)$.

Démonstration.

$(2) \Rightarrow (1)$

En effet H(A,Z) étant surjectif, à toute valeur H_m de H il correspond au moins une valeur A_i de A et une valeur k de Z telles que :

$$H_m = H(A_i,k).$$

Définissons la fonction G(H,B,Z) de la façon suivante :

$$G(H_m,B_l,k) = F(A_i,B_l,k)$$

pour toute valeur H_m de H, pour toute valeur B_l de B et pour toute valeur k de Z telle que H(A,k) puisse prendre la valeur H_m (en particulier pour la valeur A_i de A) ; cette construction de G(H,B,Z) est possible grâce à (2). Alors :

$$G[H(A_i,k),B_l,k] = F(A_i,B_l,k)$$

pour toutes valeurs A_i, B_1, k de respectivement A, B et Z, soit

$$G[H(A,Z),B,Z] = F(A,B,Z)$$

pour tous A, B et Z.

Inversement (1) $\Rightarrow$ (2). En effet

$$\begin{aligned} F(A_i,B,k) &= G[H(A_i,k),B,k] \\ &= G[H(A_j,k),B,k] \\ &= F(A_j,B,k) \end{aligned}$$

Etudions maintenant les décompositions multiples, échelonnées et en chaîne, c'est-à-dire toutes les configurations possibles de deux décompositions d'une même fonction.

II - DECOMPOSITIONS MULTIPLES

Théorème 1.

Une condition nécessaire et suffisante pour que

(1) $F(A,B,C,Z) = N[H(A,Z), M(B,Z),C,Z]$

est qu'il existe des fonctions G et K telles que

(2) $F(A,B,C,Z) = G[H(A,Z),B,C,Z]$

(3) $F(A,B,C,Z) = K[A, M(B,Z),C,Z]$

Démonstration.

Que ce soit une condition nécessaire est évident.

Montrons qu'elle est suffisante, c'est-à-dire que, d'après le lemme 1,

$$\begin{cases} H(A_i,k) = H(A_j,k) \Rightarrow F(A_i,B,C,k) = F(A_j,B,C,k) \\ M(B_i,k) = M(B_j,k) \Rightarrow F(A,B_i,C,k) = F(A,B_j,C,k) \end{cases}$$

entraîne

$$\{H(A_i,k), M(B_i,k)\} = \{H(A_j,k), M(B_j,k)\} \Rightarrow F(A_i,B_i,C,k) = F(A_j,B_j,C,k).$$

C'est évident.

Remarque.

Ce théorème est l'extension aux décompositions non disjointes, aux variables arbitraires et aux fonctions générales du théorème 2 de [KA].

III - DECOMPOSITIONS ECHELONNEES

Pour relier des décompositions d'une fonction échelonnées entre elles, il est nécessaire, comme l'a fait [KA] pour les décompositions disjointes, de distinguer dans l'ensemble des décompositions G[H(A,Z),B,Z] d'une fonction F(A,B,Z), les décompositions strictes ou décompositions G[H(A,Z),B,Z] vérifiant :

$$H(A_i,k) = H(A_j,k) \iff F(A_i,B,k) = F(A_j,B,k).$$

Quand Z est vide et quand A prend un nombre fini de valeurs, les décompositions strictes vérifient les propriétés équivalentes suivantes :

1 - $F(A_i,B) = F(A_j,B) \Rightarrow H(A_i) = H(A_j)$

2 - Card dom H est le nombre de colonnes distinctes de $\mathcal{C}^F_{B,A}$ (voir la remarque 1 de 1-4),

3 - pour toute décomposition disjointe K[L(A),B] de F(A,B) : Card dom L $\geq$ Card dom H.

Théorème 2.

Si

(1) $F(A,B,C,Z) = G[H(A,Z),B,C,Z]$

(2) $F(A,B,C,Z) = V[W(A,B,Z),C,Z]$

et si la décomposition (2) est stricte, alors il existe une fonction M telle que :

(3) $W(A,B,Z) = M[H(A,Z),B,Z]$;

donc

$$F(A,B,C,Z) = V\{M[H(A,Z),B,Z],C,Z\} .$$

Démonstration.

$W(A,B,Z) = M[H(A,Z),B,Z]$ est, d'après le lemme 1, équivalent à :

$$H(A_i,k) = H(A_j,k) \Rightarrow W(A_i,B,k) = W(A_j,B,k).$$

Or (1) implique

$$H(A_i,k) = H(A_j,k) \Rightarrow F(A_i,B,C,k) = F(A_j,B,C,k)$$

et (2) strict implique

$$F(A_i,B,C,k) = F(A_j,B,C,k) \Rightarrow W(A_i,B,k) = W(A_j,B,k).$$

Remarque 1.

Ce théorème est l'extension aux décompositions non disjointes, aux variables quelconques et aux fonctions générales du théorème 3 de [KA].

Remarque 2.

Si la décomposition $F(A,B,C,Z) = G[H(A,Z),B,C,Z]$ est stricte, alors la décomposition $W(A,B,Z) = M[H(A,Z),B,Z]$ est stricte et inversement.

En effet, (1) et (3) sont strictes si, respectivement :

$$F(A_i,B,C,k) = F(A_j,B,C,k) \Rightarrow H(A_i,k) = H(A_j,k) ,$$

$$W(A_i,B,k) = W(A_j,B,k) \Rightarrow H(A_i,k) = H(A_j,k) .$$

Or (2) stricte implique

$$F(A_i,B,C,k) = F(A_j,B,C,k) \Leftrightarrow W(A_i,B,k) = W(A_j,B,k)$$

Remarquons que si $W(A,B,Z) = M[H(A,Z),B,Z]$ est une décomposition stricte, alors $F(A,B,C,Z) = G[H(A,Z),B,C,Z]$ est stricte, que (2) soit stricte ou non.

IV - DECOMPOSITIONS EN CHAINE

Entre deux décompositions d'une fonction échelonnées entre elles, on n'a pu établir un lien que si l'une des deux au moins était stricte ; de même envisager deux décompositions en chaîne d'une fonction ne peut être intéressant que si elles vérifient une certaine propriété.

Définitions

Une fonction F(A,B) sera dite injective par rapport à A s'il existe une valeur B_o de B telle que $F(A,B_o)$ est injectif (pour tout A où $F(A,B_o)$ est défini).

Dorénavant, nous considérerons souvent des décompositions injectives ou décompositions G[H(A,Z),B,Z] vérifiant les conditions équivalentes suivantes :

1. G(H,B,Z) est injectif par rapport à H pour chaque valeur k de Z,
2. il existe pour chaque valeur k de Z, une valeur B_k de B telle que $G(H,B_k,k)$ est injectif,
3. si G[H(A,Z),B,Z] = F(A,B,Z), il existe, pour tout k, une valeur B_k de B telle que, en désignant par A_i et A_j des valeurs arbitraires de A :

$$F(A_i,B_k,k) = F(A_j,B_k,k) \Rightarrow H(A_i,k) = H(A_j,k),$$

4. il existe, pour tout k, une valeur B_k de B et une injection μ_k de dom H(A,k) dans dom G telle que :

$$G[H(A,k),B_k,k] = \mu_k \; H(A,k).$$

Nous devrons aussi considérer des fonctions F(A,B,Z) surjectives par rapport à A, pour toute valeur k de Z, c'est-à-dire telles qu'il existe, pour tout k, une valeur B_k de B entraînant

$$\text{dom } F(A,B_k,k) = \text{dom } F(A,B,k)$$

Remarque 1.

Une décomposition injective est une décomposition stricte.

En effet, si G[H(A,Z),B,Z] = F(A,B,Z) est une décomposition injective, il existe, pour toute valeur k de Z, une valeur B_k de B telle que :

$$F(A_i,B_k,k) = F(A_j,B_k,k) \Rightarrow H(A_i,k) = H(A_j,k)$$

alors

$$F(A_i,B,k) = F(A_j,B,k) \Rightarrow F(A_i,B_k,k) = F(A_j,B_k,k) \Rightarrow H(A_i,k) = H(A_j,k)$$

et

G[H(A,Z),B,Z] est une décomposition stricte.

L'inverse n'est évidemment pas vrai.

Lemme 2.

Si

$$(1) \quad \begin{cases} F(A,B,C,D,Z) = G[H(A,B,Z),C,D,Z] \\ \qquad\qquad\qquad = K[A,\ M(B,C,Z),D,Z] \end{cases}$$

et s'il existe, pour chaque valeur k de Z, des valeurs A_k et D_k de respectivement A et D telles que $K(A_k,M,D_k,k)$ est injectif et $H(A_k,B,k)$ est surjectif

(exactement dom $H(A_k,B,k)$ = dom $H(A,B,k)$), alors il existe des fonctions V et W telles que :

(2) $F(A,B,C,D,Z) = V[W(A,B,C,Z),D,Z]$,
$V(W,D_k,k)$ est injectif pour tout k,
dom $W(A,B,C,Z) \subseteq$ dom $F(A,B,C,D,Z)$,
$W(A,B,C,k) = F(A,B,C,D_k,k)$ pour tout k.

Démonstration.

Posons

$$W(A,B,C,k) = F(A,B,C,D_k,k)$$

pour toute valeur k de Z et démontrons (2) où, d'après le lemme 1, si A_i et A_j, B_i et B_j, C_i et C_j sont des valeurs arbitraires de respectivement A, B et C :

$$F(A_i,B_i,C_i,D_k,k) = F(A_j,B_j,C_j,D_k,k) \Rightarrow F(A_i,B_i,C_i,D,k) = F(A_j,B_j,C_j,D,k)$$

En effet, $H(A_k,B,k)$ surjectif entraîne l'existence de la valeur B_{ik} de B telle que :

$$H(A_k,B_{ik},k) = H(A_i,B_i,k)$$

d'où (1) entraîne :

$$\begin{aligned} F(A_i,B_i,C_i,D_k,k) &= G[H(A_i,B_i,k),C_i,D_k,k] \\ &= G[H(A_k,B_{ik},k),C_i,D_k,k] \\ &= K[A_k,\ M(B_{ik},C_i,k),D_k,k] \end{aligned}$$

De même il existe une valeur B_{jk} de B telle que :

$$\begin{cases} F(A_j,B_j,C_j,D_k,k) = K[A_k,\ M(B_{jk},C_j,k),D_k,k] \\ H(A_k,B_{jk},k) = H(A_j,B_j,k). \end{cases}$$

Alors

$F(A_i,B_i,C_i,D_k,k) = F(A_j,B_j,C_j,D\ ,k)$

$\Rightarrow K[A_k,\ M(B_{ik},C_i,k),D_k,k] = K[A_k,\ M(B_{jk},C_j,k),D_k,k]$

$\Rightarrow M(B_{ik},C_i,k) = M(B_{jk},C_j,k)$

$\Rightarrow K[A_k,\ M(B_{ik},C_i,k),D,k] = K[A_k,\ M(B_{jk},C_j,k),D,k]$

$\Rightarrow G[H(A_k,B_{ik},k),C_i,D,k] = G[H(A_k,B_{jk},k),C_j,D,k]$

$\Rightarrow G[H(A_i,B_i,k),C_i,D,k] = G[H(A_j,B_j,k),C_j,D,k]$

$\Rightarrow F(A_i,B_i,C_i,D,k) = F(A_j,B_j,C_j,D,k).$

L'injectivité de $V(W,D_k,k)$, c'est-à-dire

$V[W(A_i,B_i,C_i,k),D_k,k] = V[W(A_j,B_j,C_j,k),D_k,k]$

$\Rightarrow W(A_i,B_i,C_i,k) = W(A_j,B_j,C_j,k)$

est immédiate puisque (2) et la définition de $W(A,B,C,Z)$ entraînent

$V[W(A,B,C,k),D_k,k] = F(A,B,C,D_k,k) = W(A,B,C,k).$

Lemme 3.

Si

$$(1')\quad \begin{cases} \underline{F(A,B,C,D,Z) = G[H(A,B,Z),C,D,Z]} \\ \qquad\qquad\qquad \underline{= K[A,M(B,C,Z),D,Z]} \end{cases}$$

et s'il existe, pour chaque valeur k de Z, une valeur B_{2k} de B telle que $M(B_{2k},C,k)$ est surjectif, alors il existe des fonctions L et N telles que :

(3) $F(A,B,C,D,Z) = L[N(A,Z),\ M(B,C,Z),D,Z]$,
$\text{dom } N(A,Z) \subseteq \text{dom } H(A,B,Z)$,
$N(A,k) = H(A,B_{2k},k)$ pour tout k.

Démonstration.

Montrons d'abord l'existence de fonctions U et N telles que :

(4) $F(A,B,C,D,Z) = U[N(A,Z),B,C,D,Z]$,
$N(A,k) = H(A,B_{2k},k)$ pour tout k.

En effet, si A_i et A_j sont des valeurs arbitraires de A, on a

$H(A_i,B_{2k},k) = H(A_j,B_{2k},k) \Rightarrow G[H(A_i,B_{2k},k),C,D,k] = G[H(A_j,B_{2k},k),C,D,k]$

$\Rightarrow K[A_i,\ M(B_{2k},C,k),D,k] = K[A_j,\ M(B_{2k},C,k),D,k]$

$\Rightarrow K[A_i,\ M(B,C,k),D,k] = K[A_j,\ M(B,C,k),D,k]$

puisque $M(B_{2k},C,k)$ est surjectif

$\Rightarrow F(A_i,B,C,D,k) = F(A_j,B,C,D,k)$

donc (4) d'après le lemme 1.

Alors le théorème 1 entraîne (3), à partir de (1') et (4).

On démontrerait semblablement :

Lemme 3'

Si

$$F(A,B,C,D,Z) = G[H(A,B,Z),C,D,Z]$$
$$= K[A, M(B,C,Z),D,Z]$$

et s'il existe, pour chaque valeur k de Z, une valeur B_{1k} de B telle que $H(A,B_{1k},k)$ est surjectif, alors il existe des fonctions S et Q telles que :

$$F(A,B,C,D,Z) = S[H(A,B,Z),Q(C,Z),D,Z],$$
$$\text{dom } Q(C,Z) \subseteq \text{dom } M(B,C,Z),$$

$$Q(C,k) = M(B_{1k},C,k) \text{ pour tout } k.$$

Lemme 4.

Si

$$F(A,B,C,D,Z) = G[H(A,B,Z),C,D,Z]$$
$$= K[A,M(B,C,Z),D,Z]$$

et s'il existe, pour chaque valeur k de Z, des valeurs A_k et D_k de respectivement A et D telles que $K(A_k,M,D_k,k)$ est injectif, alors il existe des fonctions V et P telles que :

$$M(B,C,Z) = V[P(B,Z),C,Z],$$
$$\text{dom } P(B,Z) \subseteq \text{dom } H(A,B,Z),$$

$$P(B,k) = H(A_k,B,k) \text{ pour tout } k ;$$

donc

$$F(A,B,C,D,Z) = K\{A,V[P(B,Z),C,Z],D,Z\},$$

$$K(A_k,V,D_k,k) \text{ est injectif pour tout } k.$$

Démonstration.

Pour chaque valeur k de Z, il existe une injection μ_k de dom M(B,C,k) dans $K(A_k,M,D_k,k)$ telle que :

$$K[A_k, M(B,C,k),D_k,k] = \mu_k\, M(B,C,k)$$

ou

$$G[H(A_k,B,k),C,D_k,k] = \mu_k\, M(B,C,k)$$

soit en posant $H(A_k,B,k) = P(B,k)$:

$$M(B,C,k) = \mu_k^{-1}\, G[P(B,k),C,D_k,k] \ ;$$

D_k dépendant uniquement de k, il existe donc une fonction V telle que

$$M(B,C,Z) = V[P(B,Z),C,Z].$$

Remarque 2.

En plus de l'injectivité de $K(A_k,M,D_k,k)$ pour tout k, supposons la décomposition G[H(A,B,Z),C,D,Z] stricte ; alors M(B,C,Z) = V[P(B,Z),C,Z] est une décomposition stricte.

En effet

$$\begin{aligned}
M(B_i,C,k) = M(B_j,C,k) &\Rightarrow K[A,M(B_i,C,k),D,k] = K[A,M(B_j,C,k),D,k] \\
&\Rightarrow G[H(A,B_i,k),C,D,k] = G[H(A,B_j,k),C,D,k] \\
&\Rightarrow G[H(A_k,B_i,k),C,D,k] = G[H(A_k,B_j,k),C,D,k] \\
&\Rightarrow H(A_k,B_i,k) = H(A_k,B_j,k) \\
&\Rightarrow P(B_i,k) = P(B_j,k)
\end{aligned}$$

La réunion des hypothèses des lemmes 3, 3' et 4 permet maintenant d'énoncer :

Lemme 5.

Si

$$F(A,B,C,D,Z) = G[H(A,B,Z),C,D,Z]$$
$$= K[A,M(B,C,Z),D,Z]$$

et s'il existe, pour chaque valeur k de Z, des valeurs A_k et D_k de respectivement A et D, et des valeurs B_{1k} et B_{2k} de B telles que $K(A_k,M,D_k,k)$ est injectif, $H(A,B_{1k},k)$ et $M(B_{2k},C,k)$ sont surjectifs, alors il existe des fonctions L, R, N, P et Q telles que :

(5)
$$M(B,C,Z) = R[P(B,Z),Q(C,Z),Z],$$
$$F(A,B,C,D,Z) = L\{N(A,Z),R[P(B,Z),Q(C,Z),Z],D,Z\},$$

$L[N(A_k,k),R,D_k,k]$, $R[P,Q(C_o,k),k]$ et $R[P(B_{1k},k),Q,k]$ sont injectifs pour tout k et pour toute valeur C_o de C,

$$\text{dom } N(A,Z) \subset \text{dom } H(A,B,Z),$$

$$\text{dom } P(B,Z) \subset \text{dom } H(A,B,Z),$$

$$\text{dom } Q(C,Z) \subset \text{dom } M(B,C,Z).$$

Démonstration.

$K(A_k,M,D_k,k)$ étant injectif pour toute valeur k de Z, la décomposition K[A,M(B,C,Z),D,Z] est stricte d'après la remarque 1 ; le théorème 2 et le lemme 3' entraînent l'existence des fonctions U et Q telles que :

$$M(B,C,Z) = U[B,Q(C,Z),Z],$$

donc le théorème 1 et le lemme 4 entraînent l'existence de fonctions R et P telles que :

$$M(B,C,Z) = R[P(B,Z),Q(C,Z),Z],$$

soit, avec le lemme 3, de (5).

Les injectivités de $L[N(A_k,k),R,D_k,k]$, $R[P,Q(C_o,k),k]$ et $R[P(B_{1k},k),Q,k]$ sont triviales puisque, si C_o désigne une valeur arbitraire de C :

$$L\{N(A_k,k),R[P(B,k),Q(C,k),k],D_k,k\} = F(A_k,B,C,D_k,k) = K[A_k,M(B,C,k),D_k,k],$$

$$R[P(B,k),Q(C_o,k),k] = M(B,C_o,k) = \mu_k^{-1}\, K[A_k,M(B,C_o,k),D_k,k],$$

$$R[P(B_{1k},k),Q(C,k),k] = M(B_{1k},C,k) = Q(C,k).$$

Théorème 3.

Si

$$(1)\quad \begin{cases} F(A,B,C,D,Z) = G[H(A,B,Z),C,D,Z] \\ \qquad\qquad\quad = K[A,M(B,C,Z),D,Z] \end{cases}$$

et s'il existe, pour chaque valeur k de Z, des valeurs A_k et D_k de respectivement A et D, et des valeurs B_{1k} et B_{2k} de B telles que $K(A_k,M,D_k,k)$ est injectif et telles que $H(A_k,B,k)$, $H(A,B_{1k},k)$ et $M(B_{2k},C,k)$ sont surjectifs, alors il existe des fonctions V, L, R, N, P et Q telles que :

$$(6)\quad \begin{cases} F(A,B,C,D,Z) = V[L\{N(A,Z),R[P(B,Z),Q(C,Z),Z],Z\},D,Z], \\ V(L,D_k,k),\ L[N(A_k,k),R,k],\ R[P,Q(C_o,k),k] \text{ et } R[P(B_{1k},k),Q,k] \\ \qquad \text{sont injectifs pour tout k et pour toute valeur } C_o \text{ de C.} \end{cases}$$

Démonstration.

Le lemme 2 assure l'existence de fonctions V et W telles que :

$$W(A,B,C,k) = F(A,B,C,D_k,k) \ ,$$

$$(2) \qquad F(A,B,C,D,Z) = V[W(A,B,C,Z),D,Z] \ ,$$

$$\text{dom } W(A,B,C,Z) \subseteq \text{dom } F(A,B,C,D,Z),$$

$$V(W,D_k,k) \text{ est injectif pour tout } k.$$

La décomposition (2) stricte d'après la remarque 1, et (1) entraînent, d'après le théorème 2, l'existence de fonctions T et U telles que :

$$(7) \quad \begin{cases} W(A,B,C,Z) = T[H(A,B,Z),C,Z] \\ \qquad\qquad\quad = U[A,M(B,C,Z),Z] \\ U(A_k,M,k) \text{ est injectif pour tout } k \end{cases}$$

puisque

$$U[A_k,M(B,C,k),k] = W(A_k,B,C,k) = F(A_k,B,C,D_k,k) = K[A_k,M(B,C,k),D_k,k]$$

est injectif pour tout k.

Maintenant le lemme 5 appliqué à (7) assure l'existence de fonctions L, R, N, P et Q telles que :

$$\begin{cases} W(A,B,C,Z) = L\{N(A,Z),R[P(B,Z),Q(C,Z),Z],Z\}, \\ L[N(A_k,k),R,k], \ R[P,Q(C_o,k),k] \text{ et } R[P(B_{lk},k),Q,k] \text{ sont injectifs} \\ \qquad\qquad \text{pour tout } k \text{ et pour toute valeur } C_o \text{ de } C, \end{cases}$$

d'où (6) à l'aide de (7).

Remarque 3.

Dans le cas particulier où les décompositions sont disjointes ($Z=\emptyset$) les lemmes et théorèmes qui précèdent ont évidemment une forme plus simple. Le théorème 3 devient en particulier : si

$$F(A,B,C,D) = G[H(A,B),C,D] = K[A,M(B,C),D]$$

s'il existe des valeurs A_o et D_o de respectivement A et D telles que $K(A_o,M,D_o)$ est injectif et $H(A_o,B)$ est surjectif et si H(A,B) et M(B,C) sont surjectis respectivement par rapport à A et C, alors il existe des fonctions V, L, R, N, P et Q telles que :

$$F(A,B,C,D) = V[L\{N(A), R[P(B), Q(C)]\},D],$$

V(L,D), L(N,R), R(P,Q) sont injectifs respectivement par rapport à L, R, P, et Q (pour respectivement $D = D_o$, $N = N(A_o)$, Q arbitraire et $P = P(B_o)$ si B_o est une valeur de B rendant H(A,B) surjectif).

Remarque 4.

Les hypothèses autres que (1) du théorème 3 ne sont pas symétriques en A et C ; le théorème 3 implique donc un théorème dont les hypothèses et les conclusions sont symétriques des siennes. En réunissant ces deux théorèmes, on obtient la conclusion supplémentaire suivante :

$L\{N,R[P(B_{2k},k),Q(C_k,k),k],k\}$ est injectif pour tout k, et sa symétrique.

En effet, les hypothèses ajoutées à celles du théorème 3 sont : il existe, pour chaque valeur k de Z, des valeurs C_k et D_k de respectivement C et D telles que $G(H,C_k,D_k,k)$ est injectif et $M(B,C_k,k)$ est surjectif. La première entraîne :

$$L\{N(A_i,k),R[P(B_{2k},k),Q(C_k,k),k],k\} = L\{N(A_j,k),R[P(B_{2k},k),Q(C_k,k),k],k\}$$

$$\Rightarrow \quad F(A_i,B_{2k},C_k,k) = F(A_j,B_{2k},C_k,k)$$

$$\Rightarrow \quad G[H(A_i,B_{2k},k),C_k,D_k,k] = G[H(A_j,B_{2k},k),C_k,D_k,k]$$

$$\Rightarrow \quad H(A_i,B_{2k},k) = H(A_j,B_{2k},k)$$

$$\Rightarrow \quad N(A_i,k) = N(A_j,k)$$

V - DEPENDANCE

V-1. Dépendance de cardinal donné.

Généralisons la notion de dépendance d'une fonction par rapport à une de ses variables, à deux alternatives (la fonction dépend, ou ne dépend pas de la variable), en la quantifiant de la façon suivante :

Définition :

F(A,B) est dit avoir une dépendance (de cardinal) n par rapport à ses variables A si et seulement si, quelles que soient les fonctions G et H vérifiant :

$$F(A,B) = G[H(A,B),B] \qquad (1)$$

on a :

$$\min_{H} \text{Card dom } H = n \qquad (2)$$

Le concept d'indépendance d'une fonction par rapport à ses variables A devient alors équivalent à une dépendance égale à 1 de cette fonction par rapport à A ; une fonction dépendante de A a une dépendance supérieure à 1, et inversement .

Remarque 1.

A étant une fonction H(A,B) particulière, la dépendance de F(A,B) par rapport à A est inférieure ou égale à Card dom A ; F(A,B) pouvant être aussi considérée comme une fonction H(A,B) particulière, la dépendance de F(A,B) par rapport à A est inférieure ou égale à Card dom F(A,B) ; soit :

$n \leq \min \{\text{Card dom A}, \text{Card dom F}\}$.

Dans le cas où n est égal à Card dom A ou à Card dom F nous dirons que la dépendance de F(A,B) par rapport à A est maximale ou encore que F(A,B) dépend au maximum du sous-ensemble A de ses variables. La notion classique de dépendance d'une fonction booléenne par rapport à une de ses variables (booléennes) est un exemple de dépendance maximale.

Remarque 2.

Dans le cas particulier où n = Card dom A, on ne peut donc pas remplacer A par une variable dont le cardinal de l'ensemble des valeurs est inférieur à celui de A. Mais, malheureusement, ce n'est pas parce que l'ordre de dépendance n d'une fonction par rapport à une de ses variables (générale ou non) X est inférieur à Card A que l'on peut toujours remplacer cette variable par une de ses valeurs (comme en algèbre de Boole) ou par une fonction ayant moins de valeurs et différente de la fonction de départ (il suffit de remarquer que l'ordre de dépendance d'une fonction booléenne par rapport à au moins deux de ses variables est généralement deux et que les fonctions booléennes sont souvent pourtant non décomposables).

On peut envisager d'augmenter le nombre des valeurs d'une fonction pour que sa dépendance par rapport à certaines de ses variables A devienne égale à Card A. C'est ainsi que la fonction de a et de b définie par
dom

		a		
		0	1	2
b	0	0	1	1
	1	1	0	1

a une dépendance de 2, alors que la fonction

		a		
		0	1	2
b	0	0	1	2
	1	1	0	1

a une dépendance égale à Card dom a = 3.

Remarque 3

F(X) a une dépendance égale à Card dom F par rapport à X.

Théorème :

de caractérisation d'une fonction dont la dépendance par rapport à une partie de ses variables est cardinal n. Une condition nécessaire et suffisante pour que la dépendance de F(A,B) par rapport à A soit n est :

$$\max_{B} \text{Card dom } F(A,B) = n \qquad (3)$$

Démonstration.

La condition (3) est suffisante car elle entraîne d'une part que toute fonction H(A,B) vérifiant (1) satisfait :

$$\text{Card dom } H(A,B) \geq n \qquad (4)$$

d'autre part qu'il existe au moins une fonction $H^n(A,B)$ vérifiant (1) satisfaisant :

$$\text{Card dom } H^n(A,B) = n.$$

En effet, soit B_o une valeur de B pour laquelle le maximum de (3) est atteint :

$$\text{Card dom } F(A,B_o) = n$$

Alors (1) devient pour B_o

$$F(A,B_o) = G[H(A,B_o),B_o]$$

d'où

$$\text{Card } F(A,B) \leq \text{Card dom } H(A,B_o)$$

$$n \leq \text{Card dom } H(A,B_o)$$

a fortiori (4)

Considérons une fonction H(A,B) de cardinal (inférieur ou) égal à n et vérifiant la condition suivante :

$$H(A_i,B_k) = H(A_j,B_k) \Rightarrow F(A_i,B_k) = F(A_j,B_k)$$

pour toutes les valeurs B_k de B ; (3) permet d'assurer qu'une telle fonction existe. Alors le lemme 1 entraîne l'existence d'une fonction G(H,B) vérifiant :

$$G[H(A,B),B] = F(A,B).$$

Inversement, montrons que si la dépendance de F(A,B) par rapport à A est de cardinal n alors :

$$\max_B \text{Card dom } F(A,B) = n.$$

En effet, (1) donne pour toute fonction $H^n(A,B)$ de cardinal n et pour $B = B_k$

$$F(A,B_k) = G[H^n(A,B_k),B_k]$$

d'où

$$\text{Card dom } F(A,B_k) \leq \text{Card dom } H^n(A,B_k) \leq \text{Card dom } H^n(A,B) = n.$$

Cette borne supérieure n de Card $F(A,B_k)$ est atteinte ; sinon, à l'imitation de ce qui vient d'être fait, on pourrait construire une fonction H(A,B) vérifiant (1) et de cardinal strictement inférieur à n.

Conséquence 1.

La dépendance d'une fonction F(A,B) par rapport à A est supérieure ou égale à la dépendance par rapport à A de ses restrictions.

En effet, soit $\mathcal{R}_T(A,B)$ une restriction de F(A,B) de dépendance n par rapport à A ; alors il existe une valeur B_o de B telle que :

Card dom $\mathcal{R}F(A,B_o) = n$.

d'où

Card dom $F(A,B_o) \geq n$.

Conséquence 2.

Si G[H(A,B,Z),C,Z] a une dépendance n par rapport à A, alors G(H,C,Z) a une dépendance supérieure ou égale à n par rapport à H.

En effet, il existe des valeurs B_o, C_o, Z_o de B, C et Z telles que :

Card dom $G[H(A,B_o,Z_o),C_o,Z]_o = n$

d'où

Card dom $G(H,C_o,Z_o) \geq n$

et G(H,C,Z) a une dépendance supérieure ou égale à n par rapport à H.

Conséquence 3.

Si G[H(A,B,Z),C,Z] a une dépendance n par rapport à A, alors H(A,B,Z) a une dépendance supérieure ou égale à n par rapport à A.

En effet, il existe des valeurs B_o, C_o, Z_o de B, C et Z telles que :

Card dom $G[H(A,B_o,Z_o),C_o,Z_o] = n$

d'où

Card dom $H(A,B_o,Z_o) \geq n$.

et H(A,B,Z) a une dépendance supérieure ou égale à n par rapport à A.

Conséquence 4

Si F(A,B,C) a une dépendance n par rapport à A, alors sa dépendance par rapport à {A,B} est supérieure ou égale à n.

En effet, il existe B_o et C_o tels que :

Card dom $F(A,B_o,C_o) = n$

donc

Card dom $F(A,B,C_o) \geq n$.

Théorème

d'équivalence entre l'injectivité par rapport à A et la dépendance de cardinal Card dom A. Etant donné une fonction F(A,B), l'existence d'une valeur B_o de B telle que $F(A,B_o)$ est injectif entraîne que la dépendance de F(A,B) par rapport à A est Card dom A. Inversement, si Card dom A est fini et si la dépendance de F(A,B) par rapport à A est Card dom A, alors il existe une valeur B_o de B telle que $F(A,B_o)$ est injectif.

Démonstration.

On a obligatoirement :

$$\max_B \text{Card dom } F(A,B) \leq \text{Card dom A}$$

borne atteinte pour $B = B_o$ puisque $F(A,B_o)$ est injectif.

L'inverse est, aussi, immédiat.

V-2. Application aux décompositions à nombre fini de valeurs.

Quand dom A est fini, les résultats précédents permettent d'énoncer qu'une condition nécessaire et suffisante pour que la dépendance d'une fonction F(A,B) par rapport à l'ensemble A de ses variables, soit Card dom A, est que l'une des conditions équivalentes suivantes soit vérifiée :

1. $\max_B$ Card dom F(A,B) = Card dom A ;

2. il existe au moins une valeur B_o de B telle que :

   Card dom $F(A,B_o)$ = Card dom A ;

3. il existe au moins une valeur B_o de B telles que $F(A,B_o)$ soit injectif ;

4. il existe au moins une valeur B_o de B et une bijection τ des valeurs de A sur les valeurs de $F(A,B_o)$ telles que :

   $F(A,B_o) = \tau A$.

On peut utiliser la notion de dépendance pour trouver directement un résultat analogue à celui fourni par le théorème 3 dans le cas particulier où les domaines de F, H et M sont finis ; donnons-en l'énoncé dans le cas plus simple où Z est vide.

Théorème 3'.

Si

F(A,B,C,D) = G[H(A,B),C,D]

= K[A,M(B,C),D]

si Card dom F = Card dom H = Card dom M = n fini et si les dépendances de F(A,B,C,D) par rapport à A, B et C sont n,

(a) alors il existe des fonctions V, L, R, N, P et Q telles que :

$F(A,B,C,D) = V[L\{N(A),R[P(B),Q(C)]\},D]$,

Card dom L = Card dom R = Card dom N = Card dom P = Card dom Q = n,

les dépendances de V(L,D) par rapport à L, de L(N,R) par rapport à N et R, de R(P,Q) par rapport à P et Q, de N(A) par rapport à A, de P(B) par rapport à B et de Q(C) par rapport à C sont maximales, égales à n ;

(b) de même, il existe des fonctions S et T telles que :

$F(A,B,C,D) = V[S\{T[N(A),P(B)],Q(C)\},D]$,

Card dom S = Card dom T = Card dom N = Card dom P = Card dom Q = n,

les dépendances de V(S,D) par rapport à S, de S(T,Q) par rapport à T et Q, de T(N,P) par rapport à N et P, de N(A) par rapport à A, de P(B) par rapport à B et de Q(C) par rapport à C sont maximales, égales à n.

Démonstration indirecte de (a), à partir du théorème 3.

La dépendance de F(A,B,C,D) par rapport à B étant n, il existe des valeurs A_o, C_o et D_o de respectivement A, C et D telles que

Card $F(A_o,B,C_o,D_o)$ = Card $G[H(A_o,B),C_o,D_o]$ = Card $K[A_o,M(B,C_o),D_o]$ = n,

ce qui entraîne avec Card H = n

$$\text{Card } H(A_o,B) = n \quad \text{ou} \quad H(A_o,B) \text{ surjectif,}$$

et avec Card M = n

$$K(A_o,M,D_o) \text{ injectif.}$$

La dépendance de F(A,B,C,D) par rapport à A étant n, il existe B_1, C_1 et D_1 tels que :

$$\text{Card } F(A,B_1,C_1,D_1) = \text{Card } G[H(A,B_1),C_1,D_1] = n,$$

et $H(A,B_1)$ est surjectif. De même, la dépendance n de F(A,B,C,D) par rapport à C, assure l'existence d'une valeur B_2 de B telle que $M(B_2,C)$ est surjectif.

Les hypothèses du théorème 3 sont donc vérifiées ; on montrerait de même que les hypothèses symétriques, obtenues en invertissant A et C, sont vérifiées. Les conclusions des remarques 3 et 4 de IV sont donc valables. Il en résulte (a) en remarquant que les cardinalités de L, R, N, P et Q sont supérieures ou égales à n puisque les dépendances de F(A,B,C,D) par rapport à A, B et C sont n, et qu'elles sont inférieures ou égales à n puisque

$$\text{dom } L \subseteq \text{dom } F,$$

$$\text{dom } R \subseteq \text{dom } M,$$

$$\text{dom } N \subseteq \text{dom } H,$$

$$\text{dom } P \subseteq \text{dom } H,$$

$$\text{dom } Q \subseteq \text{dom } M.$$

(b) s'obtient comme (a).

Remarque 4

Le théorème 3 permet de donner des résultats plus fins que ceux mentionnés dans le théorème 3'. En particulier si Card dom F est supérieur à n, les résultats (a) sont encore valables, excepté la cardinalité et la dépendance de L. L'intérêt du théorème 3' et de ses hypothèses réside en leur simplicité en termes de dépendance ; on peut d'ailleurs encore prendre les hypothèses plus simples mais plus restrictives suivantes : F(A,B,C,D) prend n valeurs distinctes, admet deux décompositions en chaîne à fonctions composantes prenant n valeurs distinctes, et la dépendance par rapport à chacune de ses variables est n.

Remarque 5.

Le théorème 3' ou son expression plus élémentaire donnée dans la remarque précédente généralise aux fonctions de variables ayant des nombres finis de valeurs un résultat de SINGER et ASHENHURST [A] [C] [KA] et confirme une conjoncture de KARP et WINOGRAD [KA], partiellement parce que, si les hypothèses du théorème 3' sont vérifiées, il n'existe pas toujours de fonctions I et J telles que

$$(1)\quad \begin{cases} F(A,B,C,D) = I[J(A,C),B,D] \\ \\ \text{Card dom } J(A,C) \leq \text{Card dom } F \end{cases}$$

comme le montre la fonction suivante vérifiant les hypothèses du théorème 3' avec n = 3 :

$$f(a,b,c) = g[h(a,b),c] = k[a,m(b,c)]$$

f(a,b,c)		b								
		a			a			a		
		0	1	2	1	1	1	2	2	2
c		1	1	1	1	1	1	1	1	1
		2	2	2	2	2	2	2	2	2
h(a,b)		0	I	II	I	I	I	II	II	II

Ce tableau montre que la dépendance de f(a,b,c) par rapport à c est 3 et que Card dom h = 3. Le tableau

f(a,b,c)		b								
		c			c			c		
		0	1	2	1	1	2	2	1	2
a		1	1	2	1	1	2	2	1	2
		2	1	2	1	1	2	2	1	2
m(b,c)		0	I	II	I	I	II	II	I	II

montre que la dépendance de f(a,b,c) par rapport à a est maximale et que Card dom m=3. Enfin

f(a,b,c)		a								
		c			c			c		
		0	1	2	1	1	2	2	1	2
b		1	1	2	1	1	2	1	1	2
		2	1	2	2	1	2	2	1	2
j(a,c)		0	I	II	III			IV		

montre que la dépendance de f(a,b,c) par rapport à b est encore maximale. Or nous constatons que Card dom j(a,c) = 5.

On peut vérifier que cette fonction f(a,b,c) ternaire est la seule, à une permutation près des valeurs et des variables, qui vérifie les conditions du théorème 3' et qui ne vérifie pas (1). Dans le contre-exemple de [T] à l'existence de (1) dans le cas des fonctions ternaires, la variable α_3 de $f(\alpha_1, \alpha_2, \alpha_3)$ peut être considérée comme une variable binaire.

VI - CONCLUSION

Les résultats qui viennent d'être obtenus concernant la structure des décompositions multiples, échelonnées ou en chaîne d'une fonction, permettent d'obtenir la proposition suivante :

Soit une fonction prenant n (nombre fini) valeurs et à dépendance par rapport à chacune de ses variables égale à n. Considérons les fonctions composantes prenant n valeurs de ses décompositions disjointes : leurs ensembles de variables forment un treillis par rapport à la réunion et l'intersection ensembliste.

Le principe de la démonstration est dans [A] ou [C] où est démontrée la structure de treillis dans le cas des décompositions disjointes simples des fonctions booléennes.

En conclusion, nous pouvons donc dire que, mise à part l'inexistence d'une certaine décomposition montrée par le contre-exemple de la remarque 5 de V-2, tous les résultats obtenus permettent de retrouver facilement ceux relatifs aux fonctions booléennes : en effet, une décomposition booléenne disjointe simple est toujours stricte si elle dépend effectivement de la fonction composante et la dépendance d'une fonction booléenne par rapport à chacune de ses variables (effective ; sinon la variable peut être supprimée) est nécessairement deux.

Parmi les fonctions à valeurs dans un certain domaine et à variables parcourant certains domaines, la proportion de celles ayant des

décompositions non triviales strictes ou vérifiant les hypothèses du théorème 3, est d'autant plus faible que ces domaines sont plus grands et que les variables sont en plus grand nombre ; [P1] et 1-II-2 montrent par exemple que la proportion des fonctions booléennes admettant au moins une décomposition disjointe simple non triviale décroît très rapidement avec le nombre des variables ; par contre les décompositions multiples ou non disjointes sont très nombreuses.

CHAPITRE 3

PREMIERS ALGORITHMES DONNANT LES PAVES MAXIMAUX D'UNE PARTIE D'UN PRODUIT DE TREILLIS DISTRIBUTIFS

I - PRELIMINAIRES.

I.1. Définitions.

Soient $G_1, G_2, \ldots, G_I$ I treillis distributifs, $\underset{i}{.}$ et $\underset{i}{+}$ les opérations borne inférieure et borne supérieure (appelées aussi respectivement intersection ou produit et réunion) de chacun d'eux (i=1,2,...,I) ; si b_i et c_i sont des éléments arbitraires de G_i, rappelons [DUBREIL, DUBREIL-JACOTIN] :

$\underset{i}{.}$ et $\underset{i}{+}$ sont associatifs, commutatifs, idempotents

$$[b_i \underset{i}{.} (b_i \underset{i}{+} c_i) = b_i,]b_i \underset{i}{+} (b_i \underset{i}{.} c_i) = b_i$$

$$b_i \underset{i}{.} (c_i \underset{i}{+} d_i) = (b_i \underset{i}{.} c_i) \underset{i}{+} (b_i \underset{i}{.} d_i)[, b_i \underset{i}{+} (c_i \underset{i}{.} d_i) = (b_i \underset{i}{+} c_i) \underset{i}{.} (b_i \underset{i}{+} d_i)]$$

Une relation d'ordre, notée $\underset{i}{\leq}$, peut être introduite dans G_i par :

$$b_i \underset{i}{\leq} c_i \iff b_i \underset{i}{.} c_i = b_i \quad [\iff b_i \underset{i}{+} c_i = c_i]$$

Soit $G = G_1 \times G_2 \times \ldots \times G_I$ l'ensemble produit |BOURBAKI| (cardinal [DUBREIL-JACOTIN, LESIEUR, CROISOT])

des treillis $G_1, G_2, \ldots, G_I$; ses éléments seront appelés des pavés. Il est muni des I opérations suivantes :

$$b \underset{i}{*} c = (b_1.c_1,\ldots,b_{i-1}.c_{i-1},b_i+c_i,b_{i+1}.c_{i+1},\ldots,b_I.c_I), \quad i=1,2,\ldots,I$$

et de la relation d'ordre, notée $\leq$, produit des relations d'ordre $\underset{i}{\leq}$; donc, pour tous pavés $b = (b_1,\ldots,b_i,\ldots,b_I)$ et $c = (c_1,\ldots,c_i,\ldots,c_I)$ de G, on a :

$$b \leq c \iff b_i \underset{i}{\leq} c_i \qquad \text{pour tout } i = 1, 2, \ldots, I.$$

Enfin, nous dirons qu'un pavé b est compatible avec un ensemble A de pavés (ou, improprement, est pavé de A) si lui-même ou un pavé qui lui est supérieur peuvent être engendrés à partir de A à l'aide des opérations $\underset{i}{*}$. Un ensemble B de pavés est dit compatible avec A si chacun de ses éléments est compatible avec A ; inversement, A sera appelé une couverture de B.

Remarquons que quand I=1, il existe une seule opération $\underset{i}{*}$ qui est $\underset{1}{+}$.

I.2. Propriétés élémentaires.

a. $\leq$ induit sur l'ensemble $\mathcal{P}(G)$ des ensembles de pavés de G, un pré-ordre : si B et C désignent des parties de G :

$B \leq C \iff$ pour tout pavé b de B, il existe un pavé c de C tel que $b \leq c$.

b. Les opérations $\underset{i}{*}$ sont isotones.

$$\underline{b \leq c \Rightarrow b * d \leq c * d}$$
$$\underline{\qquad\qquad i \qquad\qquad i}$$

Cela résulte immédiatement de :

$$b_i \leq c_i \Rightarrow b_i . d_i \leq c_i . d_i \quad \text{et} \quad b_i + d_i \leq c_i + d_i$$

où b_i, c_i et d_i désignent des éléments arbitraires de G_i.

c. La relation de compatibilité est une relation de pré-ordre dans $\mathcal{S}_{(G)}$.

d. L'ensemble produit G des treillis distributifs G_i muni de l'ordre $\leq$ produit des ordres des G_i et des opérations

borne inférieure (ou produit ou intersection) . faisant correspondre à deux pavés b et c le pavé

$$b.c = (b_1 . c_1, b_2 . c_2, \ldots, b_I . c_I)$$

borne supérieure + faisant correspondre à deux pavés b et c le pavé

$$b + c = (b_1 + c_1, b_2 + c_2, \ldots, b_I + c_I),$$

est un treillis distributif.

I.3. Restriction de pavés.

Si $b = (b_1, b_2, \ldots, b_I)$ est un pavé de G, nous appellerons restriction à b du pavé $c = (c_1, c_2, \ldots, c_I)$, le pavé

$$\mathcal{R}_b c = (b_1 \underset{1}{.} c_1,\ b_2 \underset{2}{.} c_2,\ \ldots,\ b_I \underset{I}{.} c_I)\ ,$$

c'est-à-dire la borne inférieure b . c de b et c dans le treillis produit G.

a. La restriction à un pavé b de G, $\mathcal{R}_b$, est un homomorphisme relativement aux opérations $\underset{i}{*}$:

$$\mathcal{R}_b\ (c \underset{i}{*} d) = \mathcal{R}_b c \underset{i}{*} \mathcal{R}_b d$$

C'est une conséquence immédiate d'une part de la distributivité de $\underset{i}{.}$ par rapport à $\underset{i}{+}$, d'autre part de l'idempotence, de l'associativité et de la commutativité de $\underset{i}{.}$ pour tout i :

$$\mathcal{R}_b(c \underset{i}{*} d) = b(c \underset{i}{*} d) = (bc) \underset{i}{*} (bd) = \mathcal{R}_b c \underset{i}{*} \mathcal{R}_b d$$

b. L'application "restriction à un pavé b" de G dans G est croissante :

$$c \leq d \Rightarrow \mathcal{R}_b c \leq \mathcal{R}_b d$$

C'est une conséquence immédiate de la propriété a précédente.

c. Si un pavé c est compatible avec un ensemble A de pavés, sa restriction $\mathcal{R}_b c$ est compatible avec l'ensemble des restrictions des pavés de A.

C'est une conséquence des deux propriétés précédentes.

II - UN ALGORITHME DETERMINANT LES PAVES MAXIMAUX D'UNE PARTIE D'UN PRODUIT DE TREILLIS DISTRIBUTIFS

Etant donné un ensemble fini A de pavés, nous recherchons les pavés compatibles avec A maximaux (ou premiers) relativement à $\leq$.

II.1. Algorithme de sélection et pavés maximaux.

Montrons qu'il est possible d'engendrer les pavés maximaux compatibles avec une partie A d'un produit de treillis distributifs à l'aide de l'algorithme suivant, qui sera appelé algorithme de sélection S(A) :

On part d'un ensemble A de pavés, appelés pavés de droite et écrits en file (ou ensemble totalement ordonné ou suite ou fichier ou vecteur |HARRAND|).

On sélectionne chaque pavé de droite c, c'est-à-dire qu'on le fait passer de droite à gauche en adjoignant aux pavés de droite $\{d^k\}$ leurs $*_i$ - composés avec c, $\{c *_i d^k\}$ pour tout i et pour tout $\{d^k\}$, s'ils ne sont pas inférieurs ou égaux à un pavé de la file (pavé de droite ou pavé de gauche).

On peut adjoindre à droite un pavé s'il est compatible avec A.

On peut supprimer un pavé de droite inférieur ou égal à un pavé de droite ou de gauche.

L'algorithme existe et est terminé, quand il n'y a plus de pavés à droite.

Remarquons qu'un tel algorithme S n'est pas uniquement déterminé

(en particulier à cause de l'adjonction possible de pavés à droite).

Nous appellerons ensemble de pavés d'un algorithme de sélection tout ensemble de ses files de pavés de gauche et de pavés de droite (ils seront notés (0), (1), ..., (ν),...) et restriction au pavé b d'un ensemble de pavés (ν) la réunion de l'ensemble des restrictions à b des pavés de gauche de (ν) et de l'ensemble des restrictions à b des pavés de droite de (ν) (cette restriction sera notée $\mathcal{R}_b\nu$). Plus explicitement, si (ν) a la configuration

$\{g^j\}$ $\{d^k\}$ (ν)

ensemble de pavés de gauche ensemble de pavés de droite

($\mathcal{R}_b\nu$) aura la configuration

$\{\mathcal{R}_b\ g^j\}$ $\{\mathcal{R}_b\ d^k\}$ ($\mathcal{R}_b\nu$)

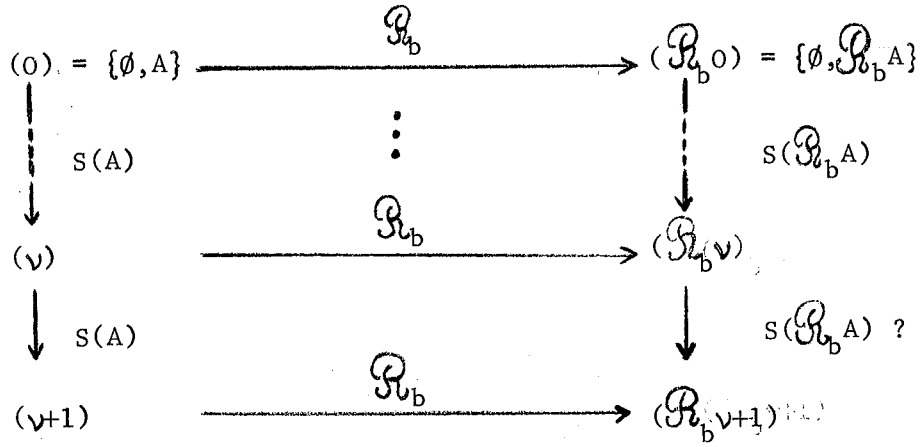
a. Lemme.

Etant donnésun algorithme de sélection S(A) appliqué à un ensemble A de pavés et un pavé b, il existe un algorithme de sélection S($\mathcal{R}_b$A) appliqué à $\mathcal{R}_b$A ayant pour suite d'ensembles de pavés les restrictions à b des ensembles de pavés de S(A).

Démonstration :

Raisonnons par induction. Nous supposons que ($\mathcal{R}_b\nu$) est obtenu à partir de ($\mathcal{R}_b$0) à l'aide d'un algorithme S ($\mathcal{R}_b$A) et que ($\mathcal{R}_b\nu$) est la restriction de (ν), c'est-à-dire : tout pavé de gauche (respectivement de droite) de ($\mathcal{R}_b\nu$) est la restriction d'un pavé de gauche (respectivement de droite) de (ν) et tout pavé de gauche (respectivement de droite) de (ν) a un pavé de gauche (respectivement de droite) de ($\mathcal{R}_b\nu$) pour restriction ; montrons que si (ν+1) s'obtient à partir de (ν) par l'une des opérations

suivantes : adjonction d'un pavé à droite ou à gauche, suppression d'un pavé de droite, il existe des opérations compatibles avec S ($\mathcal{R}_b A$) permettant de transformer ($\mathcal{R}_b \nu$) en la restriction ($\mathcal{R}_b \nu+1$) de ($\nu+1$) à b.



1. Adjonction du pavé c aux pavés de droite de (ν) : alors $\mathcal{R}_b c$, s'il n'est pas un pavé de droite de ($\mathcal{R}_b \nu$) , doit être adjoint à la file de pavés de droite de ($\mathcal{R}_b \nu$) (c'est licite d'après I-3c).

2. Adjonction du pavé c aux pavés de gauche de (ν) : alors $\mathcal{R}_b c$, s'il n'est pas pavé de gauche de ($\mathcal{R}_b \nu$) , doit être adjoint à l'ensemble de pavés de gauche de ($\mathcal{R}_b \nu$) ; chacune des adjonctions $\mathcal{R}_b c \underset{i}{*} \mathcal{R}_b d^k = \mathcal{R}_b (c \underset{i}{*} d^k)$ impliquées par la sélection de $\mathcal{R}_b c$ sera par la suite automatiquement réalisée comme restriction de l'adjonction de $c \underset{i}{*} d^k$ impliquée par la sélection de c.

3. Suppression du pavé de droite c de (ν) : alors $\mathcal{R}_b c$, s'il est un pavé de droite de ($\mathcal{R}_b \nu$) restriction du seul pavé c de (ν), doit être supprimé de l'ensemble de pavés de droite de ($\mathcal{R}_b \nu$) (c'est licite si

c est inférieur ou égal à un pavé e, puisque $\mathcal{R}_b c$ est alors inférieur ou égal au pavé $\mathcal{R}_b e$ de $(\mathcal{R}_b \nu)$; c'est aussi licite si la suppression de c est le prélude de sa sélection).

Ce lemme permet donc d'appeler la suite des restrictions à b des files de pavés de S(A) algorithme $S(\mathcal{R}_b A)$ restriction à b de l'algorithme S(A).

b. Théorème.

Soit b un pavé compatible avec A. S(A) sélectionne un pavé supérieur ou égal à b.

Démonstration par induction sur la dimension (ou somme des hauteurs [DUBREIL-JACOLIN, LESIEUR-CROISOT] de leurs composantes) des pavés. En effet, montrons que pour tout pavé b et pour toute couverture B de b formée de pavés inférieurs ou égaux à b, tout algorithme S(B), quand il est terminé, admet b comme pavé de gauche, en supposant cette propriété vérifiée pour les pavés de dimension inférieure à la dimension de b. Les pavés dont toutes les composantes ont des hauteurs au plus égales à un vérifiant cette propriété, il en résultera que les algorithmes restrictions de S(A) aux pavés b compatibles avec A, admettent ces pavés b comme pavés à gauche, donc que S(A) admet comme pavés à gauche des pavés supérieurs ou égaux aux b.

S(B) fera passer à gauche soit b, et le théorème est démontré, soit tous les pavés immédiatement inférieurs à b (pavés dont I-1 composantes sont égales aux composantes correspondantes de b et dont une est immédiatement inférieure à la composante correspondante de b) : plaçons-nous au moment où va passer à gauche un pavé immédiatement inférieur à b appelé $c = (c_1, b_2, \ldots, b_I)$ et où ne seront pas encore passés à gauche un et un

seul pavé immédiatement inférieur à b, distinct de c et ayant pour expression $e = (e_1, b_2, \ldots, b_I)$, et au plus I-1 pavés immédiatement inférieurs à b $f^{\alpha} = (b_1, b_2, \ldots, b_{\alpha-1}, f_{\alpha}, b_{\alpha+1}, \ldots, b_I)$, α parmi 2, 3, ..., I ; il y a donc à droite c et de quoi couvrir e.

On peut éliminer tout pavé de droite dont une $i^{ème}$ composante, avec i distinct de 1 et des α, est inférieure à b_i ou dont un $\alpha^{ème}$ composante est inférieure à b_{α} sans être inférieure ou égale à f_{α}. Les pavés de droite dont une $\alpha^{ème}$ composante est inférieure ou égal à f_{α} ne peuvent alors engendrer, comme nouveaux pavés, que des pavés à $\alpha^{ème}$ composante inférieure ou égale à f_{α}. Il existe donc nécessairement pour couvrir e des pavés dont chaque $i^{ème}$ composante, pour i = 2, 3, ..., I, est b_i ; la combinaison de l'un d'eux de première composante non inférieure ou égale à c_1 avec c, lors de la sélection de c, entraîne la formation de b.

II.2. Algorithme de sélection pour déterminer les pavés maximaux.

Appelons ensemble réduit d'un ensemble A de pavés l'ensemble de pavés obtenu à partir de A d'abord en ne gardant qu'un exemplaire des pavés de A, puis en supprimant les pavés inférieurs à d'autres ; il sera noté Max A.

L'algorithme de sélection S(A) devient, grâce au théorème précédent, un algorithme de recherche des pavés maximaux compatibles avec A : ce sont, après réduction, les pavés de gauche trouvés en fin d'algorithme. La réduction à tout instant de l'ensemble des pavés (et non seulement de l'ensemble des pavés de droite et de l'ensemble des pavés de gauche indépendemment l'un de l'autre) est possible à cause de l'isotonie des opérations $*_i$ et parce qu'un

pavé de droite peut jouer tout rôle joué par un pavé qui lui est inférieur ; elle parait opérationnellement intéressante.

Pour que l'ensemble de pavés de droite devienne vide, on peut en particulier :

- supprimer totalement les opérations arbitraires d'adjonction de pavés de droite ;
- ou supprimer les opérations d'adjonction de pavés de droite inférieurs ou égaux à un pavé déjà connu ; cette façon de procéder a l'avantage sur la précédente de pouvoir utiliser en cours d'algorithme une information sur l'existence d'un pavé, d'accélérer sa convergence ;
- ou effectuer des opérations d'adjonction de pavés de droite "pas trop nombreuses" vis-à-vis de la cadence de sélection ;

que l'ensemble de pavés de droite devienne vide dans les deux premières options provient de ce que les pavés compatibles avec A sont en nombre fini et de la croissance de l'ensemble de pavés inférieurs ou égaux à un pavé de gauche.

Enonçons l'algorithme de sélection en effectuant toute réduction dès qu'elle est possible:

Algorithme de sélection.

On part d'un ensemble réduit Max Â de pavés, écrit en file.

1. On "sélectionne" successivement chaque élément de la file en commençant par le premier à partir de la gauche, puis le deuxième,..., tant qu'il en existe, (en supprimant les éléments à sa gauche qui lui sont inférieurs ou égaux et) en recherchant les $*_i$ - composés qu'il admet avec les éléments à droite ; un tel $*_i$ - composé une fois formé,

- s'il est inférieur ou égal à un élément de la file, on n'en tient pas compte ;
- sinon, on supprime les éléments de la file qui lui sont inférieurs et on l'adjoint en fin de file.

2. On peut toujours en fin de file adjoindre des éléments non inférieurs ou égaux à un élément de la file (ou des éléments pas "trop nombreux") compatibles avec A.

Les pavés restants sont les pavés maximaux compatibles avec A.

Remarque 1.

La partie placée entre parenthèses de l'algorithme précédent pourra, généralement avec avantage, ne pas être utilisée.

Remarque 2.

Pour sélectionner un pavé p, il est suffisant (mais non obligatoirement intéressant) de rechercher ses $*_i$ - composés avec les éléments qui sont à sa droite lors de son transfert de l'ensemble $\{d^k\}$ de pavés de droite à l'ensemble de pavés de gauche, puisque $p *_j (p *_i d^k)$ est inférieur ou égal à p ou à $p *_i d^k$ et puisque les opérations $*_i$ sont isotones. Par contre il est nécessaire de rechercher les $*_i$ - composés de p avec les pavés éventuellement adjoints en vertu de la clause 2 de l'algorithme, s'ils ont entrainés la suppression de pavés de droite.

Remarque 3.

La possibilité d'adjoindre des pavés de droite en cours d'algorithme

n'est pas une simple généralisation, ainsi d'ailleurs que la possibilité de supprimer un pavé de droite inférieur ou égal à un pavé de gauche (voir II-1) ; elles sont nécessaires pour justifier l'algorithme de sélection.

Remarque 4.

Il peut être intéressant pour gagner de la place en mémoire et pour éviter des ruptures de séquences, de combler, aussitôt formé, le vide créé par la suppression d'un élément à droite du pavé sélectionné p (respectivement à gauche) en y portant le dernier élément de la liste ou le nouvel $*_i$ - composé introduit (respectivement le premier élément de la liste). Mieux, si un $p *_i d^k$ est supérieur à p, il peut remplacer p ; le processus de sélection de p est alors immédiatement arrêté, faisant place au processus de sélection de son remplaçant (la justification théorique est l'adjonction de $p *_i d^k$ à l'ensemble de pavés de droite).

Remarque 5.

Si on n'a d'intérêt que pour les pavés dont aucune composante n'est nulle, il est superflu de considérer au cours de l'algorithme les pavés ayant au moins une composante nulle, puisque leurs $*_i$ - composés soit sont inférieurs ou égaux à l'un des pavés qui leur a donné naissance, soit ont une composante vide.

Remarque 6.

Comme il a déjà été dit dans l'introduction, l'application à l'algèbre de Boole de l'algorithme de sélection exposé ici (abstraction faite de l'adjonction en fin de file d'éléments pas "trop nombreux"), donne l'énoncé de l'algorithme de [TISON]. Un parallèle approfondi entre monômes booléens et pavés sera

p. 4-19

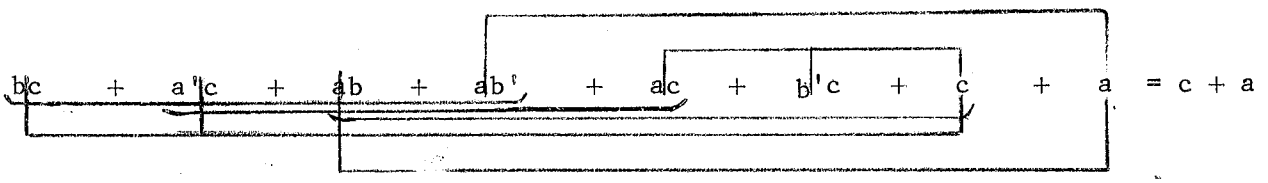
effectué dans la remarque c de 5-I-2.

Cet algorithme de recherche de monômes premiers a été programmé (voir annexe) et s'avère sur calculatrice numérique moins rapide que l'algorithme CF2 (voir 5-I-2c) de recherche de monômes premiers par consensus effectués successivement par rapport à chacune des variables, contrairement à ce qu'on pourrait penser. Cette anomalie s'explique par une disposition particulière des monômes qu'utilise [BENZAKEN] dans le programme de CF2 : pour calculer les consensus par rapport à une variable x d'un ensemble de monômes, sont envisagés les seuls couples du produit de l'ensemble de monômes contenant x et de l'ensemble de monômes contenant x' ; ainsi la considération de nombreux couples de monômes est remplacée par un tri initial des monômes en ceux contenant x , ceux contenant x' et ceux ne contenant ni x ni x' .

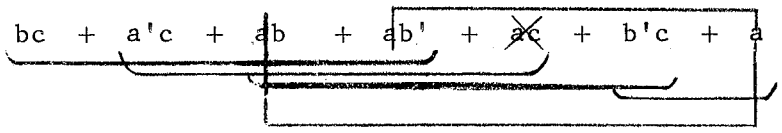
Dans le cas général (produit de treillis distributifs à plus de quatre éléments), la comparaison doit être avantageuse à l'algorithme de sélection.

Remarque 7.

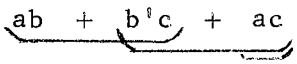
Une amélioration de l'algorithme de sélection semble difficile puisqu'il n'est pas possible d'envisager la suppression de pavés de droite compatibles avec les pavés de droite restants, comme le montre l'exemple suivant en algèbre de Boole :



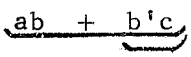
Les accolades indiquent les ensembles de monômes de droite successifs avant la sélection de leur monôme de tête ; les traits inférieurs indiquent la réduction de l'ensemble de monômes de gauche, et les traits supérieurs la réduction de l'ensemble de monômes de droite. La suppression de ac , pourtant compatible avec a'c + ab + ab' , donne :



- puisqu'il n'est pas possible d'envisager dans le cas général la recherche d' $*_i$ - composés pour certains i seulement, comme le montre l'exemple suivant en algèbre de Boole :



Le calcul des consensus par rapport à a de ab et par rapport à toutes les lettres des monômes suivants donne :



III - AUTRES PROPRIETES D'UN PRODUIT DE TREILLIS DISTRIBUTIFS

De nombreuses propriétés intéressant l'algèbre de Boole peuvent être étendues aux produits de treillis distributifs, par exemple celles relatives au calcul d'ensembles irredondants de pavés maximaux (l'extension est immédiate) et celles relatives à la dualité (voir 7-V) ; nous en donnons tout de suite quelques autres.

III.1. Généralités. Cas de treillis de Boole.

a. Notations.

Rappelons qu'un élément b_i d'un treillis G_i est dit $+_i$ - irréductible s'il admet aucune décomposition de la forme $b_i = c_i +_i d_i$ avec les inclusions strictes $c_i <_i b_i$ et $d_i <_i b_i$. De même dans le treillis distributif G produit des treillis distributifs G_i (voir I-2 d) un élément de G ou pavé b sera dit + - irréductible s'il n'admet aucune décomposition de la forme $b = c + d$ avec $c < b$ et $d < b$. Il est immédiat qu'une condition nécessaire et suffisante pour qu'un pavé soit + - irréductible est que chacune de ses composantes soit $+_i$ - irréductible.

1_i, $i = 1, 2, \ldots, I$ désignera le plus grand élément ou élément unité du treillis G_i, 0_i son plus petit élément appelé aussi élément nul ou zéro, 1 le plus grand élément de G : $1 = (1_1, 1_2, \ldots, 1_I)$. $\cup$ désignera la réunion (ensembliste) de pavés, ce qui est contraire à l'usage en algèbre de Boole, mais ce qui permet d'éviter la confusion avec la borne supérieure + .

b. Premières propriétés.

Lemme 1.

Une condition nécessaire et suffisante pour qu'un pavé + - irréductible soit compatible avec un ensemble de pavés est qu'il soit inférieur ou égal à l'un de ces pavés.

Démonstration :

La condition est évidemment suffisante. Montrons qu'elle est nécessaire.

Un élément inférieur ou égal à la borne inférieure de deux éléments b_i et c_i d'un treillis G_i est évidemment inférieur ou égal à b_i et c_i.

Si un élément $+_i -$ irréductible e_i d'un treillis distributif G_i est inférieur ou égal à la borne supérieure $b_i +_i c_i$, alors e_i est inférieur ou égal à b_i ou c_i. En effet, $e_i \cdot_i (b_i +_i c_i) = e_i \cdot_i b_i +_i e_i \cdot_i c_i$ entraîne $e_i = e_i \cdot_i b_i +_i e_i \cdot_i c_i$; soit $e_i = e_i \cdot_i b_i$ ou $e_i = e_i \cdot_i c_i$.

Il en résulte qu'un pavé + - irréductible e inférieur ou égal à l' $*_i$ - composé de deux pavés b et c, est inférieur ou égal à b ou c.

Le lemme 1 s'en déduit immédiatement par induction.

Lemme 2.

Tout pavé maximal compatible avec un ensemble A de pavés peut être obtenu, à l'aide des opérations $*_i$ (i = 1, 2, ..., I), à partir des pavés + - irréductibles inférieurs ou égaux à un pavé de A.

Démonstration .

Soit $b = (b_1, b_2, \ldots, b_I)$ un pavé maximal compatible avec A ; il est borne supérieure de pavés + - irréductibles, compatibles avec A. Soient

e_1 la première composante d'un de ces + - irréductibles : $e_1 \leq b_1$

e_2 la deuxième composante d'un de ces + - irréductibles : $e_2 \leq b_2$

.....

e_I la $i^{ème}$ composante d'un de ces + -irréductibles : $e_I \leq b_I$

$(e_1, e_2, \ldots, e_I)$ est un + - irréductible compatible avec A.

Montrons que les pavés tels que $(e_1, e_2, \ldots, e_I)$ engendrent b à l'aide des opérations $*_i$: appliquer $*_1$ à ces pavés ayant même deuxième, troisième, ..., $I^{ème}$ composantes fournit des pavés ayant pour première composante b_1 ; appliquer $*_2$ à ces pavés de première composante b_1 ayant même troisième, ..., $I^{ème}$ composantes fournit des pavés ayant pour leurs deux premières composantes respectivement b_1 et b_2 ; ... ; enfin appliquer $*_I$ donne b.

c. Cas où les treillis G_i sont booléens.

Le treillis G est alors booléen et les composantes des pavés + - irréductibles sont des atomes (ou éléments immédiatement supérieurs à l'élément nul [DUBREIL, DUBREIL-JACOTIN]) ou sont nulles.

Suivant la remarque 5 de II-2, faisons abstraction des pavés ayant une composante nulle : un ensemble A de pavés détermine une partie F des pavés atomiques (ou pavés + -irréductibles à composantes non nulles) de G , ceux inférieurs ou égaux à un pavé de A ; les pavés compatibles avec A sont les pavés dont les pavés atomiques appartiennent à F et les pavés maximaux (par rapport à $\leq$) compatibles avec A peuvent être interprétés comme "les

plus grands pavés inclus dans A". Des illustrations parlantes de ces considérations sont d'une part l'algèbre de Boole où les pavés atomiques sont appelés monômes canoniques, d'autre part les produits de seulement deux treillis booléens : 1 et les pavés sont alors représentables par des rectangles, A détermine une partie de ce rectangle.

Remarquons que les pavés qui n'ont pas d'existence graphique (pavés dont une composante au moins est nulle) sont nécessaires à la théorie faite dans ce chapitre. Par exemple considérons une algèbre de Boole à trois variables a, b et c :

$$\mathcal{R}_{a'}\ (ac \underset{a}{\ast} a'b') = \mathcal{R}_{a'}a'b' = a'b'c$$

$$\mathcal{R}_{a'}\ ac \underset{a}{\ast} \mathcal{R}_{a'}\ a'b' = (0,\ 1,\ c) \underset{a}{\ast} a'b' = a'b'c$$

que (0, 1, c) soit différent de (0, 1, 1) ou de (0, 0, 0) est nécessaire à l'application de I-3a, mais contredit l'usage de l'algèbre de Boole ; cette anomalie est sans effet car pratiquement on ne porte intérêt qu'aux monômes booléens ayant une existence graphique.

III-2. Un autre algorithme pour reconnaître si des pavés couvrent le plus grand élément d'un produit de treillis distributifs.

Lemme 3.

Si $b \cup \{c^j\}$ *couvre le plus grand élément 1 de G et si b a pour* $i^{ème}$ *composante* b_i *différent de* 1_i, *alors l'ensemble des pavés* c^j *et des* $\underset{i}{\ast}$ *- composés* $b \underset{i}{\ast} c^j$ *non inférieurs ou égaux à b couvre 1.*

Démonstration :

Le lemme 2 assure que le pavé 1 peut être obtenu à l'aide de $\underset{i}{*}$ à partir des pavés + -irréductibles inférieurs ou égaux à b ou à un c^j (nous dirons plus simplement pavés + - irréductibles de b ou d'un c^j) ; montrons que tout pavé + - irréductible de b $e = (e_1, e_2, \ldots, e_I)$ $(e_i \leq b_i$, $i = 1, 2, \ldots, I)$ est pavé + - irréductible d'un c^j non inférieur ou égal à b ou d'un $b \underset{i}{*} c^j$ non inférieur ou égal à b (avec c^j de $b \underset{i}{*} c^j$ non inférieur ou égal à b).

Soit d_i un élément + - irréductible de G_i qui n'est pas inférieur ou égal à b_i ; le pavé

$$d = (e_1, \ldots, e_{i-1}, d_i, e_{i+1}, \ldots, e_I)$$

est pavé + - irréductible d'un c^j appelé $c^o = (c_1^o, c_2^o, \ldots, c_I^o)$:

si $e_i \underset{i}{\leq} c_i^o$, e est pavé de c^o comme d ;

si $e_i \underset{i}{\not\leq} c_i^o$, e et d sont pavés de $b \underset{i}{*} c^o$.

Remarque 1.

Il est nécessaire que b_i soit différent de 1_i, car, par exemple en algèbre de Boole, l'application de la conclusion du lemme à a + a'b + a'b' donnerait a'b + a'b' ≠ 1

Remarque 2.

L'application de ce lemme à l'algèbre de Boole est à peu près le lemme 2-5 de [TISON] page 2-14.

Donnons comme application un algorithme pour savoir si un ensemble de pavés couvre 1 :

Algorithme.

On part d'un ensemble réduit de pavés écrits en file et rangés
- d'abord par ordre croissant de leurs types ; un pavé est dit de type i si i est le numéro de sa première composante différente de l'unité (ses premières composantes sont $1_1, 1_2, \ldots, 1_{i-1}$, sa $i^{ème}$ composante est inférieure à 1_i) ;
- ensuite, à l'intérieur de chaque type, par ordre croissant de leurs classes, sachant qu'une classe est un ensemble de pavés de même type i et ayant une $i^{ème}$ composante identique et sachant que l'ordre des classes de type i est l'ordre des $i^{ème}$ composantes des pavés de ces classes.

On itère, tant que c'est possible ou tant que 1 n'a pas été trouvé, les opérations suivantes : on élimine le pavé en tête de file et on calcule les $*_i$ - composés qu'il admet avec les pavés de la file du même type i que lui, mais qui ne sont pas de la même classe ou de classes supérieures : un tel $*_i$ - composé une fois formé,

- s'il est inférieur ou égal à un pavé de la file, on n'en tient pas compte ;
- sinon, on l'adjoint à la file dans le type et la classe appropriée et on supprime les pavés de la file qui lui sont inférieurs (et qui le précèdent dans la file).

On trouve 1 si et seulement si l'ensemble des pavés de départ couvre 1.

Démonstration .

D'une part le lemme 3 assure que l'ensemble des pavés en file couvre constamment 1 au cours de l'algorithme si et seulement si les pavés de départ couvre 1 . D'autre part, les $*_i$ - composés adjoints à la file ne pouvant être inférieurs ou égaux à un pavé déjà éliminé, la convergence de l'algorithme est assurée.

Remarque 3.

Il est superflu de considérer les pavés de départ dont une au moins des composantes est nulle (et différente de l'unité) conformément à la remarque 5 de II-2, et, dans le cas où un 1_i admet un seul élément immédiatement inférieur, les pavés de départ dont la $i^{ème}$ composante n'est pas 1_i .

Remarque 4.

Les remarques 2 et 4 relatives à l'algorithme de sélection (II-2) sont encore valables.

Remarque 5.

Cet algorithme est plus efficace que l'algorithme de sélection (à chaque sélection, il ne faut calculer que les $*_i$ - composés pour un certain i au lieu des $*_i$ composés, pour tout i) mais moins général.

Remarque 6.

L'application de cet algorithme aux monômes booléens est particulièrement simple puisque le nombre de classes possibles par type est au plus deux .

Considérons la somme de monômes b'c + a'c' + ab + ac' + a'bc et

l'ordre a, b, c pour les variables ; alors

monômes contenant	monômes ne contenant ni a, ni a' mais b ou b'	monôme ne contenant ni a, ni a' ni b, ni b' mais c ou c'

type :

classes :

ab + ac' + a'c' + a'bc + bc' + bc + b'c + c + c' + 1

consensus par rapport à a

consensus par rapport à b

consensus par rapport à c

Considérons maintenant la somme de monômes

x'y' + xz" + x'z + xy

et l'ordre x, y, z pour les variables

types :

classes :

xz' + xy + x'y' + x'z + yz + y' z'

Consensus par rapport à x

consensus par rapport à y

Cette somme n'est pas égale à 1.

III.3. Autres propriétés.

Etant donné un ensemble A de pavés, nous nous sommes intéressés à l'ensemble A^* des pavés (inférieurs ou égaux à des pavés) engendrés par A à l'aide des opérations $*_i$ et plus particulièrement à ses éléments maximaux Max A^*. Le lemme 2 assure que l'ensemble $Ir(A^*)$ des pavés + - irréductibles inférieurs ou égaux à A, donc l'ensemble Max $Ir(A^*)$ des pavés + - irréductibles maximaux en vertu de l'isotonie des $*_i$, permet d'engendrer à l'aide des $*_i$ l'ensemble Max A^* des pavés maximaux compatibles avec A. D'après le lemme 1, Max $Ir(A^*)$ est d'ailleurs un ensemble irredondant pour engendrer A^*.

Comme en algèbre de Boole, on peut donc attacher à un ensemble A de pavés une fonction $f_A(x_1, x_2, \ldots, x_I)$ définie sur Ir G, ensemble des pavés + - irréductibles de G, et à valeurs dans $\{0,1\}$, égale à 1 pour tous les pavés + - irréductibles inférieurs ou égaux à A, nulle ailleurs. Un élément de A^* sera appelé un pavé de f_A, un élément de Max A^* un pavé maximal ou premier de f_A, un ensemble de pavés caractéristique de f_A une base de f_A, Max A^* sa base maximale (complète), $Ir(A^*)$ sa base irréductible complète, Max $Ir(A^*)$ sa base irréductible (irredondante).

Lemme 4.

Si $\{a^j\}$ est la base maximale d'une fonction f_A, alors, si b désigne un pavé de G, $\{\mathcal{R}_b\ a^j\}$ est, après réduction, la base maximale de $\mathcal{R}_b\ f_A$.

Démonstration .

Soient a^o et a^1 deux pavés a^j arbitraires. $a^o *_i a^1$ étant inférieur ou égal à un a^j, I-3b entraîne que sa restriction, donc $\mathcal{R}a^o *_i \mathcal{R}a^1$ d'après I-3a, est inférieure ou égale à un élément de Max $\{\mathcal{R}_b\ a^j\}$. Max $\{\mathcal{R}_b\ a^j\}$ est bien base maximale complète.

Exemple.

$\mathcal{R}_b$ (ab+bc+ca) a pour base maximale a + c.

Remarque.

Ce lemme sera généralisé en 7-V-2a.

Lemme 5.

Si a est pavé maximal obligatoire d'une fonction f et si b est un pavé supérieur ou égal à a, alors $\mathcal{R}_b$ a est un pavé maximal obligatoire de $\mathcal{R}_b$ f.

La démonstration est immédiate.

Lemme 6.

Soit b un pavé dont la $i^{ème}$ composante b_i est différente de 1_i ; alors tout pavé compatible avec l'ensemble de pavés b $\cup \{a^j\}$, de borne inférieure avec b nulle ou de $i^{ème}$ composante non inférieure ou égale à b_i, est compatible avec la réunion des pavés a^j et $b *_i a^j$ non inférieurs ou égaux à b

Démonstration :

Si c est un pavé compatible avec $b \cup \{a^j\}$ de borne inférieure nulle avec b , le lemme est évident.

Si c est un pavé compatible avec $b \cup \{a^j\}$ de $i^{ème}$ composante non inférieure ou égale à b_i, I-3c assure que c est couvert par $\mathcal{R}_c b \cup \{\mathcal{R}_c a^j\}$. Les $i^{èmes}$ composantes de $\mathcal{R}_c b$ et de c étant différentes, le lemme 3 entraîne : c est couvert par la réunion des pavés $\mathcal{R}_c a^j$ et $\mathcal{R}_c a^j \underset{i}{*} \mathcal{R}_c b = \mathcal{R}_c (b \underset{i}{*} a^j)$ non inférieurs ou égaux à $\mathcal{R}_c b$, donc a fortiori par la réunion des pavés a^j et $b \underset{i}{*} a^j$ qui ne sont pas inférieurs ou égaux à b.

Lemme 6'.

Tout pavé compatible avec $b \cup \{a^j\}$ est inférieur ou égal à b ou compatible avec la réunion des pavés a^j et $b \underset{i}{*} a^j$, $j = 1, 2, \ldots, I$, non inférieurs ou égaux à b.

Démonstration :

Soit c un pavé compatible avec $b \cup \{a^j\}$

- ou c est inférieur ou égal à b ;
- ou c a une composante, soit la $i^{ème}$, non inférieure ou égale à la composante correspondante de b ; le lemme 6 assure que c est compatible avec la réunion des pavés a^j et $b \underset{i}{*} a^j$ non inférieurs ou égaux à b, donc a fortiori avec les pavés a^j et $b \underset{i}{*} a^j$, $i = 1, 2, \ldots, I$, non inférieurs ou égaux à b.

Conséquence 1.

Tout pavé maximal compatible avec $b \cup \{a^j\}$ est b ou pavé maximal compatible avec la réunion des pavés a^j et $b \underset{i}{*} a^j$, $i = 1, 2, \ldots, I$, non inférieurs ou égaux à b. Inversement, tout pavé maximal compatible avec la réunion des pavés a^j et $b \underset{i}{*} a^j$, $i = 1, 2, \ldots, I$, non inférieurs ou égaux b est un pavé maximal compatible avec $b \cup \{a^j\}$ s'il n'est pas inférieur ou égal à b (sinon, un tel pavé serait inférieur à un pavé compatible avec $b \cup \{a^j\}$ qui ne soit pas compatible avec la réunion des c^j et $b \underset{i}{*} c^j$, $i = 1, 2, \ldots, I$: il ne pourrait être qu'inférieur ou égal à b).

Conséquence 2.

Si b n'est pas compatible avec la réunion des pavés a^j et $b \underset{i}{*} a^j$, $i = 1, 2, \ldots, I$, non inférieurs ou égaux à b, alors il est pavé maximal obligatoire compatible avec $b \cup \{a^j\}$.

CHAPITRE 4

ALGORITHMES EN FILE POUR RECHERCHER LES ELEMENTS MAXIMAUX DE STRUCTURES ALGEBRIQUES

I - NOTATIONS, HYPOTHESES I ET B, CONVENTIONS.

I-1. Hypothèse d'isotonie sur une algèbre universelle ordonnée ou hypothèse I. Une conséquence pour la caractérisation des éléments maximaux.

a. Soit G un ensemble avec une relation d'ordre (partiel) notée $\leq$ et un système d'opérations algébriques n-aires (les nombres n entiers positifs peuvent être, pour différentes opérations, soit différents, soit identiques) et isotones (c'est-à-dire, si n = 1, croissantes) $* = \{*_i ; i \in I\}$. O désignera le plus petit élément et l'élément nul ($O *_i b = b *_i O = O$, $\forall b \in G$, $\forall i \in I$) de G (s'il manquait, on l'adjoindrait par commodité).

Les opérations $*_i$ seront supposées partout définies et univoques, ce qui n'est pas une restriction mais une convention algébriquement commode. En effet, dansl'optique de la recherche des éléments maximaux, qu'un $*_i$-composé soit non défini, ou qu'il soit inférieur ou égal à un élément déjà connu

ou en particulier qu'il soit nul, est équivalent (les occurrences correspondantes de $*_i$ seront d'ailleurs toutes dites inutiles) ; une opération partiellement définie et multivoque o peut donc être envisagée comme un ensemble totalement ordonné d'opérations toujours définies et univoques, en particulier nulles là où leurs valeurs ne sont pas définies par la connaissance de l'opération o.

Pour simplifier l'exposition, nous supposons dorénavant, excepté dans 6-III-3 c2 et 7-I, les opérations binaires (l'extension à des opérations non binaires ne pose pas de problème particulier). Nous avons donc, pour $i \in I$ et pour tous éléments b, c, d de G :

$$b \leq c \Rightarrow b *_i d \leq c *_i d \quad \text{et} \quad d *_i b \leq d *_i c \qquad (I)$$

et en particulier, pour tous éléments b, c, d, e de G :

$$b \leq c \quad \text{et} \quad d \leq e \Rightarrow b *_i d \leq c *_i e \qquad (1)$$

b. Considérons l'ensemble $\mathcal{P}(G)$ des parties de G avec pour lois de composition la réunion (notée $\cup$) (associative, commutative, idempotente) et les opérations (notées $*_i$) extensions des opérations $*_i$ de G à ses parties (elles sont distributives à droite et à gauche par rapport à $\cup$), muni du pré-ordre

$B \leq C \iff$ pour tout élément b de B, il existe c de C tel que $b \leq c$

extension à $\mathcal{P}(G)$ de la relation d'ordre $\leq$ définie sur G.

La relation d'équivalence

$$B \equiv C \iff B \leq C \quad \text{et} \quad C \leq B$$

définit une partition. Toutes les parties d'une même classe d'équivalence ont le même ensemble Max A d'éléments maximaux ; inversement, toutes parties ayant même ensemble Max A d'éléments maximaux appartiennent à une même classe, celle de Max A. L'isotonie des opérations $\cup$ et $*_i$ entraîne la compatibilité de l'équivalence avec ces opérations.

La restriction de la relation de pré-ordre $\leq$ aux éléments Max A de $\mathcal{P}(G)$ est une relation d'ordre.

c. Etant donnés un sous-ensemble A de G, A^* la sous-algèbre de G engendrée par A à l'aide des opérations $*$ et étant donnée la relation d'ordre $\leq$ sur G, nous nous intéressons à la recherche des éléments maximaux de A^*, ou, nous dirons aussi, à la détermination des éléments maximaux de l'algèbre $< A, \leq, * >$, définie par la donnée d'un ensemble de départ, d'une relation d'ordre et d'un ensemble d'opérations. Une caractérisation des éléments maximaux d'une telle algèbre est la suivante :

Max A^* est une partie de A^*

- à éléments deux à deux incomparables,
- supérieure ou égale à l'$*_i$-composé de deux quelconques de ses éléments,
- supérieure ou égale à A,

et inversement.

d. Notations.

$*$ désignera tout naturellement, en plus de l'ensemble des opérations $*_i$, l'opération algèbrique, au besoin multivoque, faisant correspondre à tout couple ordonné b et c d'éléments de G l'ensemble des éléments

$$\{b *_i c \ ; \ i \in I\}.$$

La partie vide de $\mathcal{P}(G)$ sera confondue avec 0.

I-2. Hypothèses de finitude ou hypothèses B.

Ces hypothèses permettent de trouver effectivement les éléments maximaux, lors d'un calcul réel.

B1 : les opérations algèbriques $*_i$ sont en nombre fini.

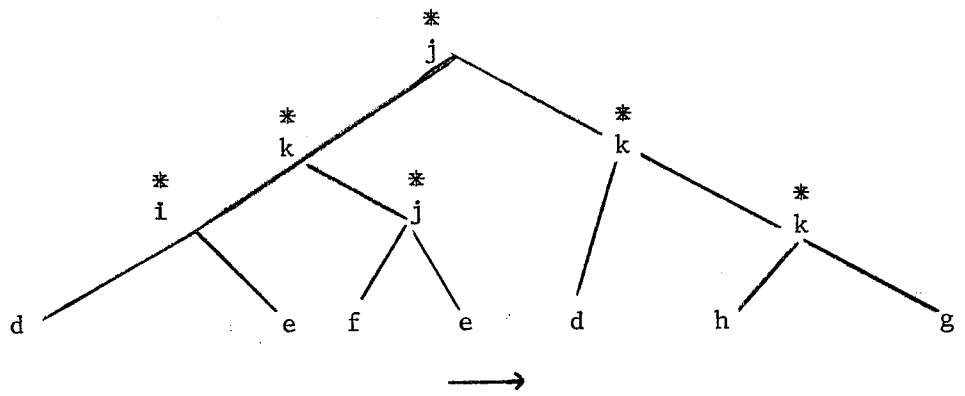
En conséquence, si, contrairement aux conventions faites en I-1 a nous considérions une opération n-aire multivoque o, les images maximales Max $\{o(b_1, b_2, \ldots, b_n)\}$ qu'elle donnerait d'un système ordonné quelconque de n éléments $b_1, b_2, \ldots, b_n$, devraient être en nombre fini.

Dorénavant, nous poserons abusivement $I = \{i ; i = 1, 2, \ldots, I\}$, c'est-à-dire I est ou un entier positif ou l'ensemble des I entiers positifs inférieurs ou égaux à I, suivant le contexte.

B2 : A est un sous-ensemble fini de B.

B3 : L'ensemble Max A^* des éléments maximaux de A^* est fini et chacun d'eux peut être obtenu à partir de A par un nombre fini d'applications des opérations $*_i$.

A tout élément maximal a de A^*, nous pouvons donc attacher au moins une arborescence (caractéristique de a, bifurcante puisque les opérations sont convenues binaires), notée $\mathcal{A}^*(A)$, dont les feuilles sont des éléments de A, donc chaque sommet, indicé par une opération $*_i$ de $*$, est un élément de A^* égal à $b *_i c$, si b et c sont les deux sommets qui le précèdent immédiatement, et dont la racine est a. Ainsi :



est l'arborescence $\mathcal{A}^{\{*,\,*,\,*\}}_{i\ j\ k}(d, e, f, g, h)$ attaché à

$$[(d \overset{i}{*} e) \overset{k}{*} (f \overset{j}{*} e)] \overset{j}{*} [d \overset{k}{*} (h \overset{k}{*} g)]$$

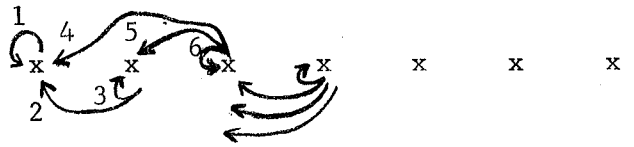
Si les opérations ne sont pas commutatives, on compose les éléments dans le sens de la flèche. Dans la suite de l'exposé, il nous arrivera d'identifier un arbre et sa racine.

I-3. Conventions.

A moins de spécification contraire, nous ferons les conventions suivantes qui précisent complètement, quand ils ne le sont pas déjà, les algorithmes donnés ultérieurement (ce qui est en particulier utile pour traiter des exemples), qui sont commodes pour les comparer , enfin qui ne restreignent les variantes utiles que d'une manière raisonnable (voir à ce propos en particulier II-3 e).

a. L'examen de tous les couples d'éléments d'une file sera ainsi fait :

On considère chaque élément (en commençant par le premier à partir de la gauche, puis le deuxième,...) et tous les éléments qui le précèdent dans la file (en commençant par le premier à partir de la gauche, puis le deuxième,..., puis lui-même)



b. Calculer les $*_1$-composés d'un couple d'éléments, c'est calculer d'abord leur $*_1$-composé, puis leur $*_2$-composé, ..., enfin leur $*_I$-composé.

c. Des éléments ajoutés à une file le seront à la fin de la file et par ordre de leur formation.

d. Quand on veut supprimer une des deux occurrences d'un même élément dans une liste, on supprime la deuxième.

II - ALGORITHMES EN FILE PRIMITIFS

II-1. Algorithmes de réduction.

Une opération essentielle dans des algorithmes de recherche d'éléments maximaux est la réduction d'une file A ou obtention de Max A, c'est-à dire d'une seule occurrence des éléments maximaux de A.

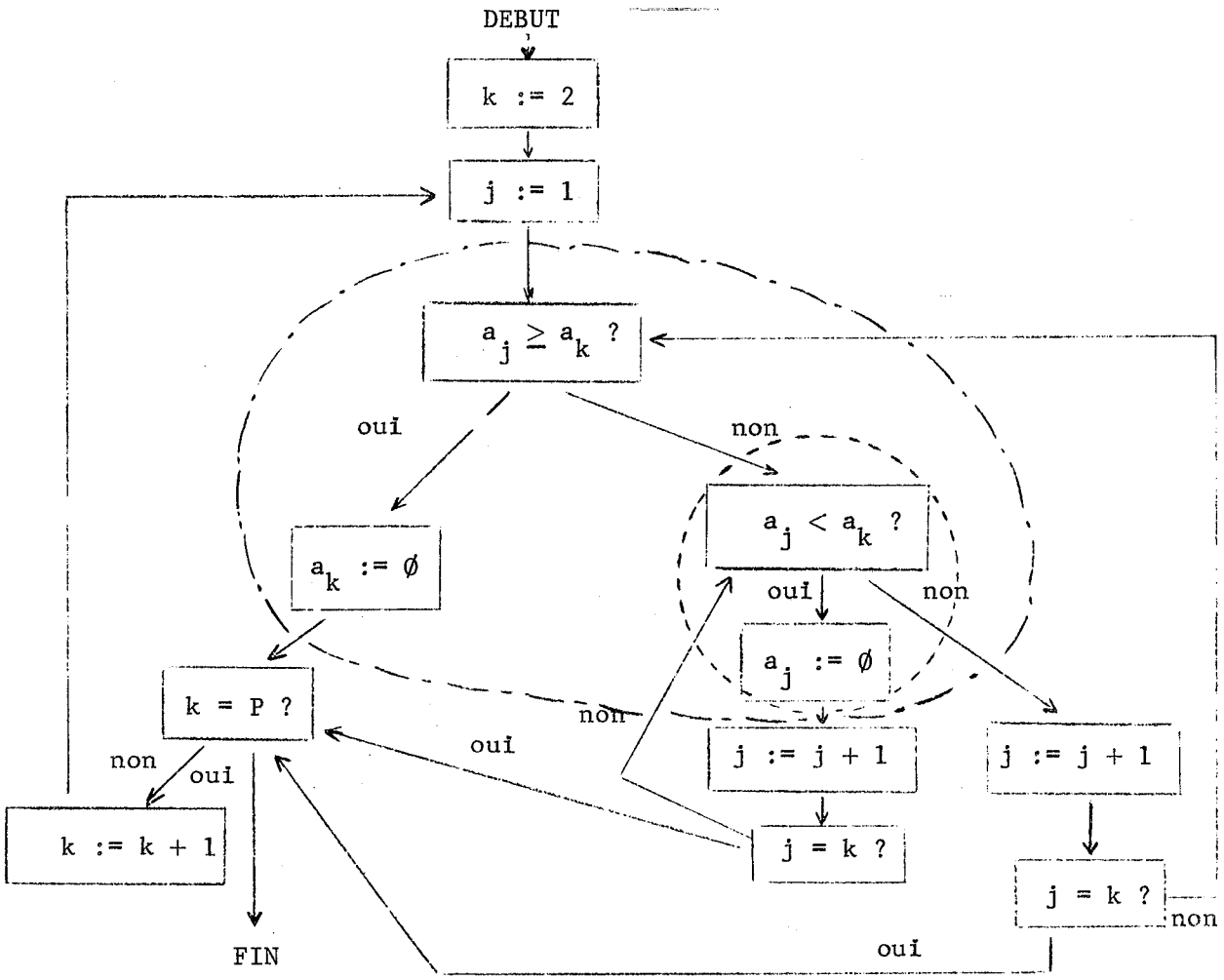
a. Algorithmes de réduction.

La convention I-3 a, en considérant les opérations $*_1$ et $*_2$ commutatives et binaires ainsi définies sur $A = \{a_1, a_2, \ldots, a_p\}$:

$$\begin{cases} a_j *_1 a_k = a_j & \text{si } a_j \geq a_k \\ *_1 \text{ non défini} & \text{sinon} \end{cases}$$

$$\begin{cases} a_j *_2 a_k = a_k & \text{si } a_j < a_k \\ *_2 \text{ non défini} & \text{sinon} \end{cases}$$

donne l'algorithme suivant de réduction :



Les données sont :

P, supposé ≥ 2 ,

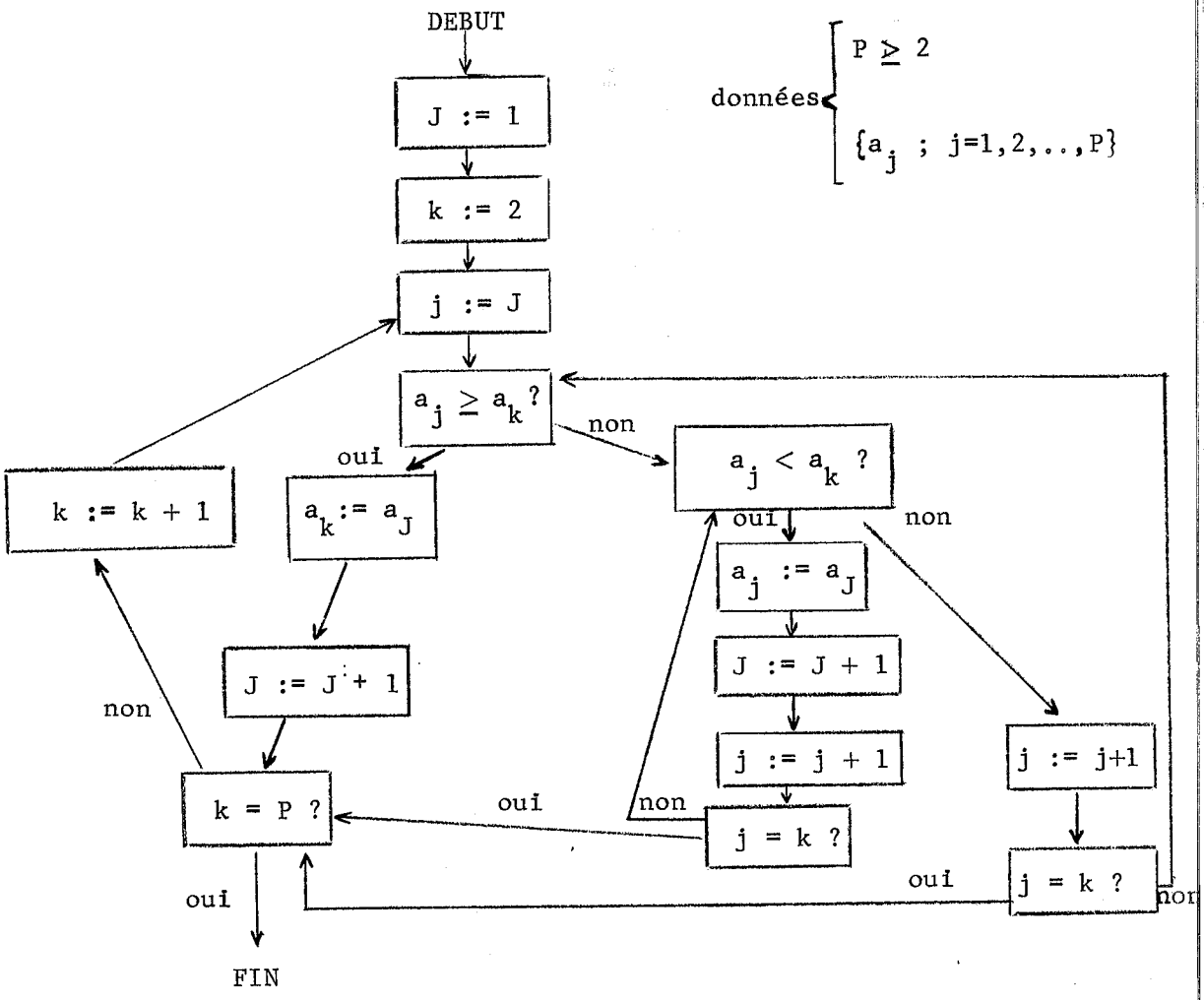
$\emptyset$, élément arbitraire strictement inférieur à tous les éléments de Max A ,

$\{a_j, j = 1, 2, \ldots, P\}$;

une liste de P éléments contenant a_j à la $j^{ème}$ place si a_j était la première occurrence d'un élément de Max A, sinon contenant $\emptyset$, est fournie. Le nombre

de couples examinés est évidemment au plus égal à $\frac{(P-1)P}{2}$.

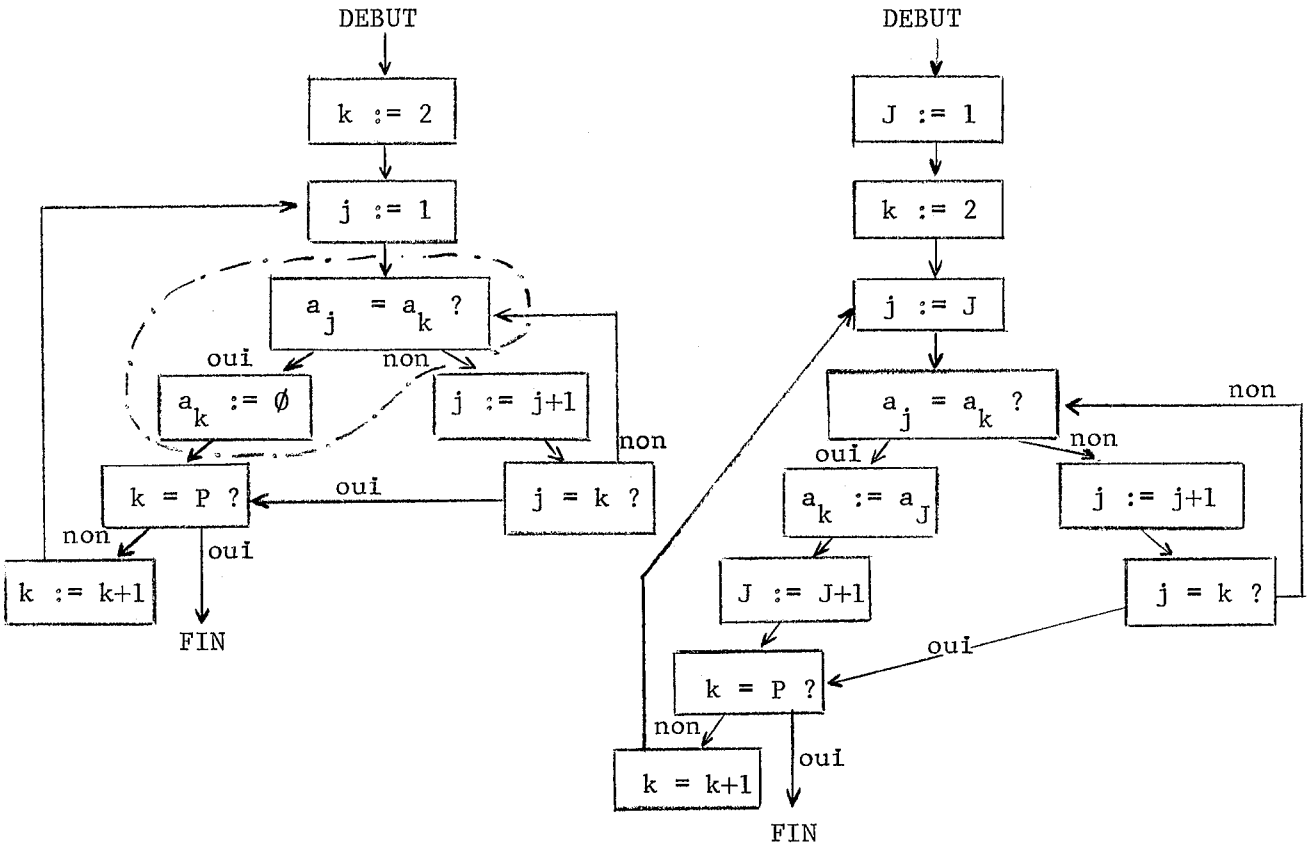
Pour obtenir Max A dans des mémoires juxtaposées, on peut opérer de la façon suivante, faisant fi de la convention I-3-a :



Le résultat est les P-J+1 éléments de Max A dans a_J, a_{J+1}, a_{J+2}, ..., a_P.

Le nombre de couples examinés est inférieur à celui de l'algorithme précédent puisqu'on ne considère pas les mémoires vides.

b. Algorithmes d'unique occurrence donnant d'un ensemble d'éléments une seule occurrence de chaque élément. Ce sont des cas particuliers des algorithmes de réduction ; considérés en eux-mêmes, ils sont aussi localement déterminés grâce aux conventions I-3a et I-3d.



II-2. Des algorithmes en file.

Le théorème de caractérisation entraîne l'existence des algorithmes suivants donnant Max A^* à partir de A ; ils sont entièrement précisés à cause des conventions précédentes (I-3) :

Algorithme F1 ou algorithme en file à tours.

A partir de A écrit en file, on obtient Max A^* en employant alternativement les deux règles suivantes :

- éliminer les éléments inférieurs ou égaux à un autre,
- ajouter globalement en fin de file les * - composés des éléments deux à deux,

jusqu'à la stabilisation du procédé.

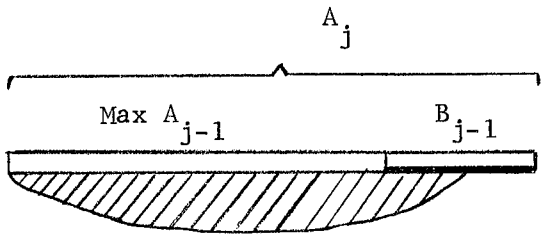
A chaque tour de l'algorithme F1, il peut être préférable de tenir compte des renseignements accumulés pendant les tours précédents. En pratique, on pourra opérer ainsi pendant le $(j+1)^{ème}$ tour, en désignant par A_j l'ensemble de départ (donc $A_o = A$) partitionné en les deux ensembles suivants fournis par le $j^{ème}$ tour :

Max A_{j-1}

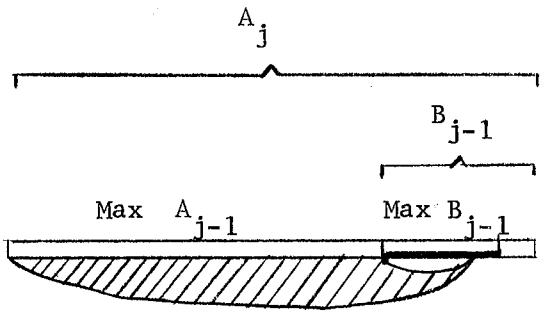
B_{j-1} ou une partie des * -composés des éléments puis deux à deux de Max A_{j-1} :

1. L'élimination des éléments non maximaux de A_j, c'est-à-dire l'obtention de Max A_j, pourra se faire :

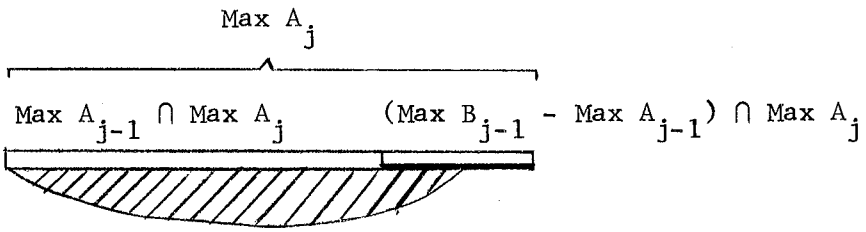
soit conformément aux algorithmes de réduction donnés en II-1 a : on supprime les éléments non maximaux de A_j = Max $A_{j-1} \cup B_{j-1}$ en considérant dans la file A_j chaque élément de B_{j-1} avec les éléments qui le précèdent, comme l'indique le schéma suivant :



soit nonobstant les conventions I-3 : on supprime d'abord les éléments maximaux de B_{j-1}, pour obtenir Max B_{j-1} ; puis on réduit Max A_{j-1} $\cup$ Max B_{j-1} en comparant chaque élément de Max B_{j-1} avec les seuls éléments de Max A_{j-1} ; cette méthode sera d'autant plus intéressante par rapport à la précédente que Card B_{j-1} est plus grand par rapport à Card Max B_{j-1}, que Card Max A_{j-1} est plus grand et que Card Max A_j est plus grand par rapport à Card Max A_{j-1}.



2. La recherche des $*$ - composés des éléments de Max A_j pris deux à deux se fait en examinant chaque élément de (Max B_{j-1} - Max A_{j-1}) $\cap$ Max A_j avec tous les éléments qui le précédent



Ce qui vient d'être dit (mise à part évidemment ce qui est en contradiction avec les conventions I-3) est partie intrinsèque de F1 ; des examens de couples faits aux tours antérieurs sont évités et remplacés par une manipulation d'index.

Algorithme F2 ou algorithme en file à pas.

On part du sous-ensemble Max A de A, écrit en file. On obtient Max A^* en recherchant les $*$ - composés de chaque élément de la file (en commençant par le premier à partir de la gauche, puis le deuxième, ... tant qu'il en existe) avec les éléments qui le précèdent dans la file (en commençant par le premier à partir de la gauche, puis le deuxième, ..., puis lui-même) ; dès qu'un $*$ - composé est formé (il en existe 2I pour tout couple d'éléments),

- s'il est inférieur ou égal à un élément de la file, on l'élimine,
- sinon, on l'ajoute en fin de file et on supprime les éléments qui lui sont inférieurs.

Algorithme F3 ou algorithme en file récursif.

Soit $A = \{a_1, a_2, \ldots, a_P\}$ ou mieux Max $A = \{a_1, a_2, .., a_P\}$.

A partir de $A_o = \emptyset$, et pour j = 1, 2,...,P, on calcule successivement :

$$A_p = \text{Max } [\text{Max } (A_{j-1} \cup a_p]^*$$

à l'aide de F2. Alors, Max $A^* = A_P$.

La stabilisation de ces algorithmes vient de ce que l'ensemble des éléments sur lequel ils travaillent, est croissant et de l'hypothèse de finitude B3.

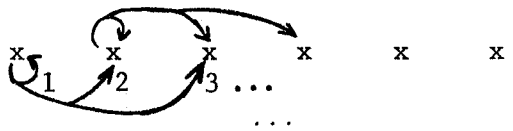
II-3. Remarques.

a. Tout en maintenant les conventions faites en I-3, on pourrait donner d'autres algorithmes de recherche des éléments maximaux ; on pourrait par exemple dans F3 utiliser F1 au lieu de F2. Ce sont en particulier les

applications qui montreront le bien-fondé du choix de F1, F2 et F3. F1 et F2 diffèrent par le moment où l'on commence les comparaisons par rapport aux combinaisons.

b. Les algorithmes F1 et F2 sont identiques quand les * - composés ne sont pas supérieurs aux éléments déjà trouvés et les trois algorithmes sont identiques quand il n'y a pas formation de nouveaux * - composés. F2 et F3 ne diffèrent que par les endroits d'insertion des * - composés : en fin de file pour F2 et dans la file avant a_{j+1} pour le calcul de A_j de F3.

c. Si, au lieu de I-3a, la convention suivante pour l'examen de tous les couples d'éléments d'une file avait été faite : on considère chaque élément (en commençant par le premier élément à partir de la gauche, puis le deuxième,...) et les éléments qui le suivent :



F2 n'apparaîtrait plus comme une sorte d'extension de convention. Un algorithme utilisant une telle progression des éléments "en avant" (explicitement : on part du sous-ensemble Max A de A, écrit en file. On obtient Max A^* en recherchant les * - composés de chaque élément de la file (en commençant par le premier à partir de la gauche, puis le deuxième,... tant qu'il en existe) avec lui-même et les éléments qui le suivent dans la file ; dès qu'un * - composé est formé,

- s'il est inférieur ou égal à un élément de la file, on l'élimine,
- sinon, on l'ajoute en fin de file et on supprime les éléments qui lui sont inférieurs) n'est valable, en

règle générale que si les * - composés formés ne peuvent pas avoir d'* - composé

avec les éléments dont l'examen est terminé (c'est le cas de chaque tour de l'algorithme F1 et c'est le cas des réductions étudiées en II-1) ; il a été aussi valable pour la recherche des pavés maximaux d'une partie d'un produit de treillis distributifs (voir chapitre 3) et il le sera pour déterminer la réunion de relations d'équivalence (5-II-2b).

d. Les algorithmes de réduction peuvent être considérés comme des algorithmes particuliers de recherche d'éléments maximaux. Les algorithmes F1 et F2 ne donnent rien, leurs ensembles de départ étant déjà Max A ; par contre, l'algorithme F3 conduit au même algorithme de réduction que la convention I-3 a.

e. Il peut être intéressant pour gagner de la place en mémoire de calculatrice et pour éviter des ruptures de séquences, de combler, aussitôt formé, le vide créé par la suppression d'un élément inférieur, en y portant le premier élément de la liste ou le dernier ou un élément du couple d'éléments examiné ou le composé de ce couple d'éléments, selon en particulier la situation du vide. Une telle façon de procéder bouscule l'énoncé mais non l'esprit des algorithmes donnés et doit vérifier l'évidence suivante : on ne peut ajouter aucun élément à la partie dont l'examen est terminé, si ce n'est un élément examiné.

Un tel remplissage des vides est par exemple utilisé dans l'algorithme de recherche des monômes premiers d'une fonction booléenne de [GILLI, MEO] ; il a aussi été explicitement réalisé pour les algorithmes de réduction (II-1) et il le sera dans 5-II-3b.

f. Cas d'un algorithme tel que toute opération $*_i$ est soit à occurrence inutile (définition en I-1 a) , soit à produit $a *_i b$ supérieur à a et b (l'opération $*_i$ est alors dite utile). Il se produit en particulier si l'ensemble est totalement ordonné.

Appelons Card A le nombre d'éléments de A, U le nombre d'opérations utiles effectuées, Card (Max A^*) le nombre d'éléments maximaux de A^*. On a:

$$\text{Card Max } A^* = \text{Card } A - U \geq 1$$

Le nombre d'opérations utiles effectuées est donc inférieur ou égal à Card A - 1

En tenant compte de la remarque e, les algorithmes en file à pas et récursif peuvent être ainsi aménagés : dès qu'un $*$ - composé de deux éléments a et b est utilement formé (a est supposé à gauche de b dans la file), d'abord on remplace b par a $*$ b, ensuite on supprime les éléments inférieurs à a $*$ b en les remplaçant par le premier élément de la file s'ils sont à gauche de a $*$ b et par le dernier s'ils sont à droite de a$*$ b. F2 et F3 ainsi modifiés sont identiques. Une application est donnée en 5-II.

III - LA CONDITION C SUR LES OPERATIONS $\overset{*}{}_i$ ET LES ALGORITHMES C

Dans un souci d'allègement des notations, nous supposons dans ce paragraphe les opérations $\overset{*}{}_i$ commutatives ; cependant le théorème et l'algorithme qui suivent sont valables quand la commutativité n'existe pas.

III-1. Théorème

Si les conditons I et B sont vérifiées et si (condition C) pour tous j, k = 1, 2, ..., I avec j < k et pour toutes arborescences $\mathcal{B}^{*k}(A^{1\,2\,\ldots\,k-1})$ et $\mathcal{C}^{*k}(A^{1\,2\,\ldots\,k-1})$ (pour la notation voir I-2), il existe une arborescence $\mathcal{D}^{*k}(A^{1\,2\,\ldots\,k-1})$ telle que :

$$\mathcal{B}^{*k}(A^{1\,2\,\ldots\,k-1}) \overset{*}{}_j \mathcal{C}^{*k}(A^{1\,2\,\ldots\,k-1}) \leq \mathcal{D}^{*k}(A^{1\,2\,\ldots\,k-1}) \qquad (C)$$

alors :

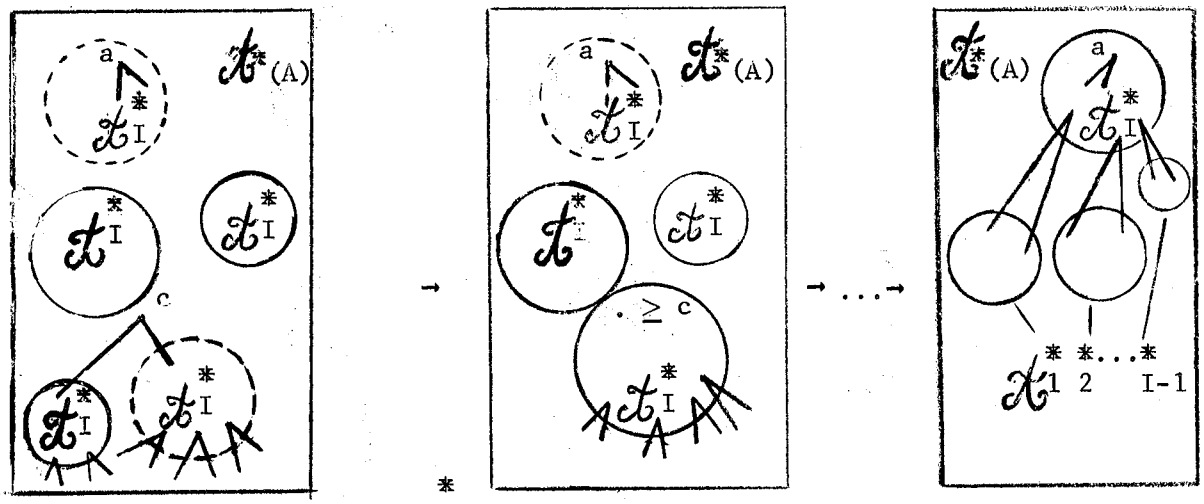
$$\text{Max } A^* = \text{Max } [(\ldots\{\text{Max } [(\text{Max } A)^{*1}]\}^{*2}\ldots)^{*I}]. \qquad (2)$$

Démonstration.

Si b est un élément quelconque de $A^{* * *...*}_{1\ 2\ \ k-1}$, une expression particulière de (C) est évidemment :

$$\mathcal{B}^{*}_{k}(A^{* * *...*}_{1\ 2\ \ k-1})\ {}^{*}_{j}\ b \leq \mathcal{D}^{*}_{k}(A^{* * *...*}_{1\ 2\ \ k-1}) \qquad (C')$$

Soient a un élément maximal arbitraire de A^*, $\mathcal{A}^*(A)$ une arborescence qui lui est attachée. Montrons d'abord qu'il existe une arborescence de racine a, indicée par $*_I$ et de feuilles appartenant à $A^{* * *...*}_{1\ 2\ \ I-1}$.



En effet appelons $\mathcal{A}^{*}_{I}$ les composantes connexes de sommets indicés par $*_I$ de $\mathcal{A}^*(A)$ (ce sont des sous-arborescentes disjointes de $\mathcal{A}^*(A)$). S'il n'y a pas qu'une seule composante $\mathcal{A}^{*}_{I}$ contenant a, il existe une composante $\mathcal{A}^{*}_{I}$ qui précède immédiatement un sommet c et telle que tous les sommets indicés par $*_I$ et précédant c appartiennent à des composantes qui

précèdent immédiatement c (il y a au plus deux telles composantes). (C) (ou (C')) permet de transformer la sous-arborescence de racine c en une sous-arborescence de racine indicée par $*_I$ et supérieure ou égale à c, et telle que les sommets indicés par $*_I$ forment une seule composante connexe ; (I) permet d'affirmer que la racine de l'arborescence reste a. On continue le procédé jusqu'à ce que (C) soit appliqué en a.

Ensuite (2) est déduit par induction sur les sous-arborescences obtenues en supprimant la composante connexe indicée par l'opération $*_I$.

III-2. Si les conditions I, B et C sont satisfaites, on déduit les algorithmes suivants donnant Max A^* à partir de A :

Algorithmes C.

On part de Max A et on recherche successivement à l'aide d'un des algorithmes en file

$$\text{Max } A^{*_1} = \text{Max } [(\text{Max } A)^{*_1}]$$

$$\text{Max } A^{*_1 *_2} = \text{Max } [(\text{Max } A^{*_1})^{*_2}]$$

....

$$\text{Max } A^* = \text{Max } [(\text{Max } A^{*_1 *_2 \ldots *_{I-1}})^{*_I}].$$

Pour chacune des I itérations, on pourra utiliser un algorithme en file (F1 ou F2 ou F3 et on appellera respectivement CF1, CF2, CF3 les algorithmes en file obtenus).

Remarque 1.

(C) est une condition suffisante d'existence de (2) ; ce n'est pas une condition nécessaire.

Remarque 2.

(C) est en particulier vérifiée quand l'opération $*_k$ pour tout $k = 1, 2, \ldots, I$, est distribuée par rapport à toutes les opérations $*_j$ avec $j = 1, 2, \ldots, k-1$, c'est-à-dire quand pour tous éléments b, c et d de G

$$\begin{cases} b *_j (c *_k d) = (b *_j c) *_k (b *_j d) \\ (c *_k d) *_j b = (c *_j b) *_k (d *_j b) \end{cases}$$

ou quand les opérations $*$ vérifient pour tous $j, k = 1, 2, \ldots, I$ et pour tous éléments b, c, d de G

$$b *_j (c *_k d) = (b *_j c) *_k d$$

(de telles opérations $*$ seront dites pseudo-associatives). On le démontre facilement par induction.

IV - FONCTION DE LOCALISATION

Les algorithmes en file exigent le calcul, pour tout couple d'éléments b et c, de $b *_i c$; or, si on sait a priori que les $*_i$ - composés de certains couples d'éléments sont inférieures ou égaux à des éléments déjà trouvés (ou sont non définis dans notre terminologie algorithmique), il est préférable de ne pas les considérer. Formalisons cette constatation.

Etant données les opérations $*_i$ sur G x G et des relations binaires R_i sur $H^K \times H^K$ (H ensemble arbitraire), nous appellerons fonction de localisation L de $*_i$ une fonction générale $\{l_k ; k = 1, 2, \ldots, K, K \text{ entier fini}\}$ définie sur G et à valeurs dans H^K, vérifiant :

$$b *_i c \text{ défini} \Rightarrow L(b)\ R_i\ L(c), \text{ pour } i = 1, 2, \ldots, I$$

La fonction de localisation (avec sa relation associée) d'opérations $*_i$ donne une borne supérieure du domaine où les $*_i$ sont utiles. Le renseignement qu'elle apporte sur la localisation des couples utiles est implicitement contenu dans G ; elle ne fait que l'expliciter. Cette information pourra être à l'origine d'un support qui évite la considération de couples inutiles et qui remplace la disposition en file des algorithmes généraux. Les algorithmes de recherche d'éléments maximaux seront généralement d'autant plus efficaces qu'ils utiliseront mieux l'information apportée par la fonction de localisation.

Comme application de la notion de fonction de localisation, introduisons les algorithmes matriciels généralisés ainsi définis : H est un ensemble fini, K = 2, R_i a la forme 1_2 (b) R'_i 1_1 (c) où R'_i est une relation binaire arbitraire sur H x H. A tout élément $a_{\alpha\beta}$ de G est donc attaché un couple ordonné d'indices (α, β), où $\alpha, \beta \in H$; et un $*_i$ - composé $b_{\alpha\beta} *_i c_{\gamma\delta}$ de deux éléments $b_{\alpha\beta}$ et $c_{\gamma\delta}$ de G sont définis seulement si $\beta R'_i \gamma$.

Nous verrons en détail au chapitre 3 les algorithmes matriciels (non généralisés), algorithmes matriciels généralisés ainsi spécifiés :

$$\begin{cases} H = I \\ \beta\ R'_i\ \gamma\ , \text{ pour } i = 1, 2, \ldots, I, \text{ est identique à } \beta = \gamma = i\ ; \end{cases}$$

non seulement leur support, comme celui des algorithmes matriciels généralisés, est une matrice, mais aussi leurs opérations $*_i$ sont compatibles avec le produit matriciel.

Comme autres applications de la notion de fonction de localisation, il y a les développements (voir à ce sujet le chapitre 7) et les monômes marqués introduits dans [KUNTZMANN 2] p.99-102 ou triplets de consensus, de lettres de marque et de variables accessoires ; dans l'ensemble des monômes marqués , l'ordre est ainsi défini : un monôme, ayant ses premier et second groupes figurant dans le premier et le second groupe d'un autre monôme, est supérieur à ce second monôme ; l'* - composé de deux monômes est le triplet du consensus des premiers groupes des deux monômes, du produit booléen des lettres de marque et du produit booléen des variables accessoires et de la variable par rapport à laquelle on a formé le consensus ; la fonction de localisation $(1_1, 1_2)$ de * définie sur les monômes marqués et à valeurs les variables de leurs consensus et leurs variables accessoires, vérifie

$$b * c \text{ défini} \Rightarrow 1_1 (b) \cap 1_2 (b) = 1_2 (b) \cap 1_1 (c) = \emptyset$$

V - REMARQUES.

Etant donnés un ensemble de départ A, une opération $*$ et une relation d'ordre $\leq$, nous avons donné des algorithmes pour la recherche des éléments maximaux de A^* et nous avons laissé prévoir l'utilisation de dispositions particulières pour les éléments permettant de traduire automatiquement les renseignements fournis par la fonction de localisation, évitant des considérations superflues de couples d'éléments.
Maintenant, nous développerons des applications. Faisons auparavant les deux observations suivantes :

V.1. Tout problème d'énumération peut-être considéré comme un problème de recherche d'éléments maximaux à la seule condition (essentielle, mais en fait peu contraignante : la relation d'ordre choisie doit être suffisemment grossière) que les éléments à énumérer soient des éléments maximaux ; la résolution d'un problème d'énumération se décompose alors en

- le choix d'un triplet $(A, \leq, *)$;
- la recherche à partir de $(A, \leq, *)$ des éléments maximaux de A^* ;
- la recherche (éventuelle) parmi ces éléments maximaux du sous-ensemble des éléments à énumérer.

V-2. Un problème délicat est le choix d'un triplet $(A, \leq, *)$ auquel appliquer les algorithmes, à cause de la corrélation entre A, $\leq$ et $*$; en effet,

- plus $*$ est simple, facilement calculé, plus les éléments sur lesquels $*$ opère doivent appartenir à des classes précises d'éléments, ce qui entraîne deux conséquences :

1. les ensembles de départ A possibles doivent remplir des conditions exigeantes : leur nombre d'éléments sera souvent important; la nature de ces éléments devra être suffisamment "atomique" pour pouvoir servir de racines à des arborescences de sommets les éléments à énumérer ; on peut dire que les A possibles devront être des ensembles complets d'éléments de mêmes caractéristiques.

2. les relations d'ordre possibles sont grossières, les opérations * devront opérer sur des catégories complètes d'éléments (avec un i
ordre plus fin certains de ces éléments pourraient être éliminés comme inférieurs à des éléments nouvellement créés) ; autrement dit, la condition I n'est vérifiée que si les relations d'ordre sont suffisamment grossières.

- plus * est complexe, plus l'ensemble de départ peut être quelconque, la relation d'ordre fine.

Une illustration de cette remarque est les méthodes de recherche des monômes premiers d'une fonction booléenne à l'aide des consensus (voir 5-I-2c ; A est alors un ensemble arbitraire de monômes caractéristiques de la fonction et la relation d'ordre est la relation d'ordre ensembliste) et la méthode de McCLUSKEY (pratiquement on part de la forme canonique et il n'y a pas de relation d'ordre entre les monômes booléens ; voir 7-III).

CHAPITRE 5

APPLICATIONS DIRECTES DES ALGORITHMES EN FILE

I - PAVES MAXIMAUX D'UNE PARTIE D'UN PRODUIT DE TREILLIS DISTRIBUTIFS

Les algorithmes du chapitre précédent permettent de déterminer très facilement des méthodes (différentes de celle exposée dans le chapitre 3) pour déterminer les pavés maximaux d'une partie d'un produit de I treillis distributifs.

Les notations et définitions sont les mêmes qu'en 3-I.

I-1. Algorithmes.

On constate :

1. La condition I est vérifiée par toute opération $*_i$ puisque, pour tous éléments b_i, c_i, d_i de G_i :

$$b_i \leq_i c_i \Rightarrow b_i *_i d_i \leq_i c_i *_i d_i \quad \text{et} \quad b_i +_i d_i \leq_i c_i +_i d_i$$

2. (B) est évidemment vérifiée ;

3. (C) est vérifiée ; plus précisément, nous avons, pour tous i,j=1,2,..., I :

$$b \underset{j}{*} (c \underset{i}{*} d) = (b \underset{j}{*} c) \underset{i}{*} (b \underset{j}{*} d)$$

puisque :

$$b_j \underset{j}{+} (c_j \underset{j}{.} d_j) = (b_j \underset{j}{+} c_j) \underset{j}{.} (b_j \underset{j}{+} d_j) \text{ , pour tout } j$$

$$b_i \underset{i}{.} (c_i \underset{i}{+} d_i) = (b_i \underset{i}{.} c_i) \underset{i}{+} (b_i \underset{i}{.} d_i) \text{ , pour tout } i$$

$$b_k \underset{k}{.} (c_k \underset{k}{.} d_k) = (b_k \underset{k}{.} c_k) \underset{k}{.} (b_k \underset{k}{.} d_k) \text{ , pour tout } k \text{ (différent de i et j).}$$

De ces propriétés, il résulte qu'on peut utiliser les algorithmes F1, F2 et F3 et les algorithmes C pour calculer, à partir de A, les pavés maximaux compatibles avec A.

I-2. Remarques.

a. L'algorithme CF1 ou l'algorithme CF2, dans le cas où les treillis G_i sont en plus complémentés, est énoncé dans [TISON 2].

b. Quand I = 2 et quand les treillis sont booléens, F1 est dans [MALGRANGE] et CF1 ou CF2 sont dans [MALGRANGE] et [CHEIN].

c. Dans le cas où chaque treillis G_i est booléen et a, en plus des éléments zéro et unité, deux et seulement deux éléments x_i et x'_i, les

pavés et les pavés maximaux compatibles avec A et dont aucune composante n'est nulle sont respectivement les monômes et les monômes premiers de la fonction booléenne $f(x_1, x_2, \ldots, x_n)$ égale à 1 sur A, nulle ailleurs.

Le $*_i$ - composé de deux monômes b et c a alors l'interprétation suivante :

- si b et c admettent x_i et seulement x_i pour lettre différemment accentuée, $b *_i c$ est leur consensus [QUINE] ;

- si b et c n'ont aucune lettre qui soit différemment accentuée ou admettent au moins une lettre distincte de x_i qui soit différemment accentuée, $b *_i c$ est un monôme multiple de b ou de c (ou inférieur ou égal à b ou c), alors que le consensus de b et c n'est pas défini [KUNTZMANN 2].

L'$*_i$ - composé de deux monômes est donc leur consensus s'il existe, sinon des multiples de ces monômes. Les monômes multiples d'un autre étant constamment supprimés dans les algorithmes F et C et compte tenu de la remarque 5 de 3-II-2, les notions de $*_i$ - composé et de consensus, de $*_i$ - composé et de consensus par rapport à la variable x_i coïncident donc pratiquement ; les algorithmes F1, F2 et CF1 ou CF2 appliqués aux fonctions booléennes deviennent des algorithmes connus de recherche des monômes premiers (on peut voir respectivement [KUNTZMANN 2], [GILLI , MEO], [TISON 1]).

d. Considérer un $*_i$ - composé de deux pavés comme non défini ou nul quand il est inférieur ou égal à l'un de ces pavés, évite la réduction de cet $*_i$ - composé et est donc algorithmiquement intéressant.

e. La remarque 5 de 3-II-2est encore valable.

I-3. Applications.

En plus des monômes premiers d'une fonction booléenne, les algorithmes de recherche de pavés maximaux permettent en particulier de trouver

a. les ensembles d'articulation (ou sous-ensembles de sommets dont la suppression disconnecte l'hypergraphe) minimaux d'un hypergraphe [BERGE 2] connexe. Par exemple, déterminons les ensembles d'articulation minimaux d'un hypergraphe à n sommets s_k (k=1,2,...,n) qui le séparent en I(= 2 ou 3 ou 4 ... ou n-1) sous-hypergraphes disjoints ; à l'hypergraphe, associons l'hypermatrice booléenne M_1 symétrique, surunitaire et à I dimensions suivante : à tout I-uplet $(s_{k_1}, s_{k_2}, \ldots, s_{k_I})$ correspond un et un seul élément de M_1 ; un élément de M_1 est 1 si et seulement si les sommets correspondants sont non tous distincts ou sont adjacents ; recherchons-en les pavés vides maximaux. Les ensembles d'articulation minimaux sont les compléments (par rapport à l'ensemble des sommets de l'hypergraphe) des sommets des composantes des pavés vides maximaux trouvés, chacun des I sous-hypergraphes disjoints ayant pour ensemble de sommets les sommets d'une composante du pavé vide maximal correspondant.

Le cas particulier où I = 2 et où l'hypergraphe est un graphe non orienté est dans [MALGRANGE] et [KAUFMANN 1].

b. les décompositions simples additives, multiplicatives et disjonctives maximales d'une fonction booléenne, et en conséquence toutes ses décompositions simplescomme il est expliqué dans [PICHAT 1]. Comme pour les

ensembles d'articulation, il peut-être intéressant de généraliser cette étude si les opérateurs somme, produit ou disjonction ont plus de deux entrées. Le cas particulier des décompositions simples disjointes est étudié en II-3.

c. les implantations ou application d'un graphe dans un graphe, c'est-à-dire les sous-graphes partiels d'un graphe isomorphes à un autre graphe [MALGRANGE], [KAUFMANN 1].

d. les pavés maximaux d'une application de Ir G (voir 3-III-3) dans un treillis distributif [PICHAT 2].

e. les ensembles fortement stables et les cliques maximaux d'un hypergraphe. En déterminant les rectangles nuls maximaux de sa matrice booléenne symétrique à diagonale nulle associée, puis l'intersection des composantes de chaque rectangle maximal, enfin les intersections maximales, on pourrait obtenir les carrés nuls maximaux. Une méthode plus rapide sera étudiée en 7-V et 7-VI.

II - REUNION DE RELATIONS D'EQUIVALENCE

II-1. Notations.

Etant donné l'ensemble fini $X = \{x_i ; i=1,2,...,I\}$ sur lequel sont définies des relations d'équivalence, nous appellerons

- G, l'ensemble des parties de X, l'ordre dans G étant l'inclusion ensembliste ($\subseteq$) ;
- A, l'ensemble des classes d'équivalence des relations d'équivalence données, à plus d'un élément ;
- $*$ - composé de deux parties de X : leur réunion si elles ont un élément commun, sinon l'ensemble vide ;
- $*_i$ - composé de deux parties de X : leur réunion si elles ont l'élément x_i en commun, sinon l'ensemble vide (i=1,2,...,I).

II-2. Algorithmes.

a. Les conditions I, B et C étant satisfaites (on a en particulier :

$a *_i (b *_j c) \subseteq b *_j (a *_i c)$ ou $c *_j (a *_i b)$ pour tous i, j = 1, 2, ..., I),

nous pouvons utiliser les trois algorithmes en file donnés, les algorithmes C et aussi la remarque f de 4-II-3.

L'algorithme récursif F3 est dans [BENZAKEN 1] p. 137 ; il demande,

ainsi que F2, au plus $\frac{\text{Card A (Card A-1)}}{2}$ opérations. Cette borne est atteinte quand $A = \text{Max } A^*$ ou quand $\text{Card } A = \text{Card Max } A^* + 1$ et quand la dernière classe de la file A a un élément commun avec une autre classe de la file.

Les algorithmes CF1 ou CF2 reviennent à réunir d'abord les classes qui contiennent x_1, puis celles qui contiennent $x_2, \ldots$, enfin celles qui contiennent x_I ; ils demandent donc au plus I(Card A-1) opérations (quand les classes de A sont les éléments de X, le nombre d'opérations est $\frac{I\ (I-1)}{2}$ car, si la $i^{\text{ème}}$ itération a fourni la classe réduite au seul élément x_i, cette classe n'a pas à être examiné au cours des itérations suivantes).

b. Donnons maintenant un algorithme dont la progression dans l'examen des couples de classes rappelle l'algorithme de sélection (chapitre 3) ; nous pourrons cependant le comparer aux algorithmes CF1 ou CF2.

Algorithme

Soit un ensemble $A = A_o = \{a_1^o, a_2^o, \ldots, a_p^o\}$ de classes d'équivalence écrit en file.

On construit une suite A_{j+1} à partir d'une suite $A_j = \{a_1^j, a_2^j, \ldots, a_Q^j\}$ $(j=0,1,2,\ldots)$ de la façon suivante :

1. Si a_1^j est élément de A, on l'élimine de la file et on forme successivement, pour $q = 2,3,\ldots,Q$ et $b_1 = a_1^j$:

$$\begin{cases} b_q = b_{q-1} & \text{si} \quad b_{q-1} \cap a_q^j = \emptyset \\ b_q = b_{q-1} \cup a_q^j & \text{sinon} \end{cases}$$

et dans ce cas, a_q^j est éliminé de la file.

Soit b_p le terme b_q formé de la réunion de termes adjacents de A^j qui est de rang le plus élevé, c'est-à-dire :

$$\begin{cases} b_p = a_1^j \cup a_2^j \cup \ldots \cup a_p^j \\ \text{avec} \quad b_p \cap a_{p+1}^j = \emptyset \ . \end{cases}$$

Alors A_{j+1} est A_j abstraction faite des termes éliminés et après adjonction de b_Q en fin de file si $b_p \neq b_Q$; si $b_p = b_Q$, b_Q est sélectionné en début de file et A_{j+1} est A_j abstraction des termes éliminés.

2. Si a_1^j n'est pas élément de A ou si A_j est vide, l'algorithme est terminé.

Les classes d'équivalence maximales sont les termes de la file obtenue en fin d'algorithme s'il y en a et les termes sélectionnés.

Exemple 1.

$$A = bg + de\ell + cg + hi + bc + a + jk + bc + e\ell + ij$$

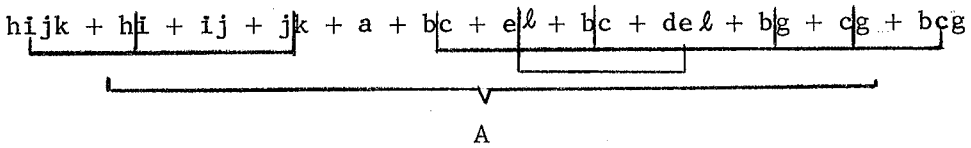
$$\text{Max } A^* = bg + de\ell + cg + hi + bc + a + jk + bc + e\ell + ij + bcg + hij + hijk$$

Les classes de départ a_1^j sont successivement bg, deℓ , hi, a, jk ; les b_p correspondants sont bg, deℓ , hi, a, jk et les b_Q : beg, deℓ , hij, a, hijk ; deℓ et a sont donc sélectionnés, pratiquement laissés en tête de file.

Les classes d'équivalence maximales sont donc : deℓ , a, bcg, hijk.

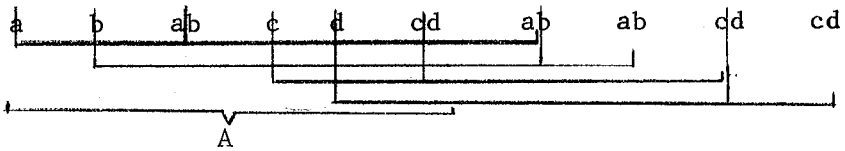
Exemple 2.

Considérons l'ensemble de départ de l'exemple 1 mais avec un ordre différent :



Les a_1^j sont successivement hi, a, bc, eℓ, les b_p : hijk, a, bc, deℓ, et les b_Q : hijk, a, bcg, deℓ ; hijk, a et deℓ sont donc placés en tête de file.

Exemple 3.



Les classes d'équivalence maximales sont ab et cd.

Démonstration.

Les réunions d'éléments adjacents b_p égales à b_Q peuvent être sélectionnées puisqu'elles sont disjointes des classes de la file résultante ; quand une telle réunion b_p est sélectionnée, aucun de ses éléments ne se trouve plus dans la file, et quand une réunion b_Q est mise en fin de file, b_Q est la seule classe de la file contenant les éléments de la classe b_p correspondante.

Il est suffisant de prendre pour classes de départ a_1^j les seules classes de A ; en effet, considérons un élément x_1 et la première classe de

la file A le contenant, soit ce sera une classe de départ, soit sa réunion sera réalisée avec une classe de départ et dans les deux cas on effectuera la réunion de toutes les classes contenant x_i. D'après l'algorithme CF1 ou CF2, on sait que l'on obtient ainsi les classes maximales.

Remarque 1.

Le nombre d'itérations (la $i^{ème}$ fournit A_j à partir de A_{j-1}) est au plus I, puisqu'une classe de départ a_1^j a tous ses éléments distincts des éléments des classes de départ précédentes. Il y a donc au plus I classes qui sont adjointes à la file au cours de l'algorithme, chaque classe de départ donnant naissance à au plus une classe b_Q . L'exemple 3 montre que ces bornes supérieures sont atteintes.

D'ailleurs la $I^{ème}$ classe de départ, si elle existe, est constituée d'un seul élément x_j ; sa considération permet de l'éliminer ou d'être sûr qu'elle est classe maximale.

Remarque 2.

Savoir si deux classes ont des éléments communs n'est souvent pas plus long actuellement sur calculatrice électronique que savoir si elles ont un élément commun donné x_i. L'algorithme de sélection est alors meilleur que CF1 ou CF2 ; il est meilleur dès qu'une classe de départ a_1^j a plus d'un élément : en effet, si x_{i_1} , x_{i_2} , ..., x_{i_p} sont ses éléments, il effectue simultanément au moins les opérations $*_{i_1}$, $*_{i_2}$, ..., $*_{i_p}$.

c. Quelques remarques.

1. Dans tous les algorithmes précédents, les opérations de réduction ne sont pas faites indépendamment de la recherche de nouvelles classes, mais en même temps ; on évite ainsi des énumérations supplémentaires de couples de classes.

2. Il est inutile de considérer les classes à un seul élément si on ne veut pas vérifier qu'il est effectivement un élément d'une classe de A.

3. Si des renseignements supplémentaires sont connus sur la relation d'équivalence réunion cherchée, on peut essayer de choisir un algorithme plus rapide.

Par exemple, vérifier que Card A classes d'équivalence sont deux à deux disjointes demande seulement (Card A-1) opérations ainsi définies sur un couple de classes (le premier couple considéré est deux classes quelconques de A) :

* voir d'abord si les deux classes opérandes sont disjointes ;
* si elles le sont, faire leur réunion, et itérer le processus avec cette réunion et une classe de A non encore considérée ; si elles ne le sont pas, les classes de A ne sont pas disjointes.

On peut améliorer tous les algorithmes précédents en choisissant convenablement l'ordre des classes de A. En règle générale, il est commode de placer en début de file peu de classes ayant de nombreux éléments, non disjointes et couvrant X, de façon à obtenir rapidement la relation d'équivalence réunion ; il sera préférable de placer en tête les classes ayant le plus d'éléments.

4. La remarque e de 4-II-3, pour combler les vides créés par les éliminations, est toujours valable.

II-3. Applications.

Ces algorithmes sont des moyens très efficaces de trouver la fermeture transitive d'une matrice booléenne symétrique comme nous le verrons en 6-III-3 d4 (A est alors un ensemble de classes de deux éléments au plus) et d'un graphe symétrique, c'est-à-dire ses sous-graphes connexes maximaux.

Nous détaillons ici l'application à la recherche de la décomposition disjointe simple additive [PICHAT 1] d'une fonction booléenne.

Par exemple, considérons la fonction :

$$f(a,b,c,d,e,g,h,i,j,k,\ell) = b'g+de\ell+c'g+hi+bc'+a+jk+b'c+e'\ell'+ij$$

donnée par l'ensemble de ses monômes premiers m.

A chaque monôme m, faisons correspondre une relation d'équivalence comme suit : deux variables sont équivalentes si et seulement si m les contient. La réunion de ces relations d'équivalence définit une partition qui est aussi la partition de la décomposition disjointe simple additive de f. Nous avons donc :

$$A = bg + de\ell + cg + hi + bc + a + jk + bc + e\ell + ij$$

$$\text{Max } A = bg + de\ell + cg + hi + bc\ (+a) + jk + ij$$

a. F1 donne :

$$bg + de\ell + cg + hi + bc + jk + ij + (bcg + bcg + bcg + hij + ijk)$$

Nous simplifions et continuons

$$de\ell + bcg + hij + ijk + (hijk)$$

Nous simplifions

$$de\ell + bcg + hijk$$

Cette dernière expression est stable.

b. F2 donne

$$bg + de\ell + cg + hi + bc + jk + ij$$

$$de\ell + hi + jk + ij + bcg$$

$$de\ell + jk + bcg + hij$$

$$de\ell + bcg + hijk$$

F2 modifié à l'aide des remarques e et f de 4-II-3 donne :

$$bg + de\ell + cg + hi + bc + jk + ij$$

$$de\ell + bcg + hi + ij + jk$$

$$bcg + de\ell + hij + jk$$

$$de\ell + bcg + hijk$$

c. F3 donne

$$bg + de\ell + cg$$

$$de\ell + bcg$$

Rajoutons hi, puis bc

$$de\ell + bcg + hi + bc$$

Rajoutons jk

$$de\ell + bcg + hi + jk$$

Rajoutons ij

$$de\ell + bcg + hi + jk + ij + hij + hijk$$

d. Maintenant utilisons par exemple l'algorithme CF2 avec l'ordre b, c, d, e, g, h, i, j, k, ℓ pour les opérations ; on considère donc d'abord les classes qui contiennent b, puis celles qui contiennent c, ..., enfin celles qui contiennent ℓ .

$$\text{Max } A = bg + de\ell + cg + hi + bc + jk + ij$$

$$\text{Max } A^{*b} = bg + de\ell + cg + hi + bc + jk + ij + bcg$$

$$\text{Max } A^{**bc} = de\ell + hi + jk + ij + bcg = \text{Max } A^{*bcdegh}$$

$$\text{Max } A^{*******bcdeghi} = de\ell + hi + jk + ij + bcg + hij$$

$$\text{Max } A^{*********bcdeghij} = de\ell + jk + bcg + hij + hijk = \text{Max } A^{**********bcdeghijk}$$

$$\text{Max } A^{*} = de\ell + bcg + hijk$$

Il peut être intéressant de ne pas réduire à la fin de chaque itération ; ainsi à la $i^{ème}$ itération seules les classes contenant la lettre i seraient supprimées pour donner une nouvelle classe mi ; les classes inférieures ou égales à m ne seraient pas supprimées.

On peut aussi utiliser la remarque f de 4-II-3 pour simplifier.

e. L'application de l'algorithme de sélection est donnée dans l'exemple 1 de 2b.

f. La partition de la décomposition disjointe simple additive de $f(a, b, c, d, e, g, h, i, j, k, \ell)$ est donc

$$\boxed{(a,\ bcg,\ de\ell,\ hijk)}$$

et sa décomposition disjointe simple additive

$$f(a,b,c,d,e,g,h,i,j,k,\ell) = (a) + (b'g+c'g+bc'+b'c) + (de\ell+e'\ell') + (hi+jk+ij).$$

III - AUTRES APPLICATIONS DIRECTES

D'autres applications directes des algorithmes généraux en file seront données ultérieurement, en particulier pour :

1. calculer la fermeture transitive d'une matrice définie sur un pseudo-treillis distributif (voir 6-III-3d) ;

2. effectuer les développements (voir chapitre 7).

CHAPITRE 6

ALGORITHMES MATRICIELS

Ce chapitre traite d'ensembles $< A , \leq , * >$ vérifiant en plus des conditions d'isotonie I et, sauf spécification contraire, de finitude B, les deux hypothèses suivantes ou condition M :

1. à chacun de leurs éléments $a_{\alpha\beta}$ est attaché un couple ordonné d'indices (α,β), où $\alpha,\beta \in \{1, 2, \ldots, I\}$,

2. les $*$ - composés $b_{\alpha\beta} * c_{\gamma\delta}$ de deux éléments $b_{\alpha\beta}$ et $c_{\gamma\delta}$ sont définis seulement si $\beta = \gamma$,

ou, autrement dit, d'ensembles G sur lesquels existe une fonction de localisation $L = \{l_1, l_2\}$ appliquant G dans I x I (I ensemble fini), la relation R ayant la forme $l_2(b) = l_1(c)$ (notations de 4-IV).

Pour alléger les définitions, le plus petit élément sera aussi bien désigné par $0_{\alpha\beta}$ (avec $\alpha,\beta = 1, 2, \ldots, I$) que par 0 .

Cette condition permet d'utiliser une disposition matricielle des éléments et d'exclure, grâce à une sélection systématique des couples d'éléments au niveau des cases de la matrice, la recherche de certaines $*$ - compositions inutiles. Les algorithmes en file du chapitre 4 deviennent alors des

algorithmes matriciels qui seront étudiés en I ; II donne des applications de ces algorithmes matriciels ; enfin III montre que ces algorithmes matriciels permettent de trouver la fermeture de certaines matrices.

Aux conventions de 4-I-3, nous ajouterons la suivante :
les produits matriciels seront faits en commençant par le couple 1ère ligne-1ère colonne, puis 1ère ligne-2ème colonne, ..., puis 2ème ligne-1ère colonne, ...

I - ALGORITHMES MATRICIELS GENERAUX

I.1. Algorithmes matriciels à tours et à pas.

a. Algorithmes matriciels à tours.

1. L'algorithme F1 et la condition M donnent

Algorithme M1 ou algorithme matriciel à tours.

On part du sous-ensemble Max A de A, mis sous forme matricielle M(Max A) de la façon suivante : la case (α, β) est l'ensemble des éléments $\{\alpha_{\gamma\delta} \mid \gamma = \alpha , \delta = \beta\}$ de Max A. On emploie alternativement les deux règles suivantes :

1. pour $\alpha, \beta = 1, 2, \ldots, I$, on effectue le produit matriciel de la $\alpha^{\text{ème}}$ ligne avec la $\beta^{\text{ème}}$ colonne, noté avec [FALKOFF, IVERSON, SUSSENGUTH] $\alpha^{\text{ème}}$ ligne $\overset{\cup}{*}$ $\beta^{\text{ème}}$ colonne (on cherche les composés suivant $*$ des éléments de la case (α, i) avec les éléments de la case (i, β), pour $i = 1, 2, \ldots, I$) ;

2. on les place globalement à leurs justes cases dans la matrice s'ils ne sont pas inférieurs ou égaux à des éléments de la matrice et on supprime les éléments de la matrice qui, leurs sont inférieurs ; jusqu'à la stabilisation du procédé. Max A est l'ensemble des éléments de la matrice stationnaire.

Le nombre d'opérations de (1) au niveau des cases matricielles est I^3 (au lieu de, dans un algorithme en file, I^4 ou, en supposant la commutativité au niveau des cases, $\frac{I^2 (1 + I^2)}{2}$; ces deux dernières estimations de

coût sont des bornes supérieures car des cases matricielles peuvent être vides et ne demander aucun traitement si on comble les vides comme l'indique la remarque 4-II-3e).

2. Dans le cas particulier où deux éléments non nuls d'indices différents sont incomparables et où les $*$ - composés de deux éléments $b_{\alpha i}$ et $c_{i\beta}$ ont pour indices associés (α, β) c'est-à-dire sont de la forme $d_{\alpha\beta}$ (conditions D), l'algorithme M1 peut-être exprimé de la façon suivante :

Algorithme M1'

A partir de l'écriture matricielle M(Max A) de Max A, on calcule à l'aide de la formule de récurrence

$$\begin{cases} N_2 j = \text{Max}\,[N_2 j\text{-}1 \cup (N_2 j\text{-}1 \overset{\cup}{*} N_2 j\text{-}1)] \\ j = 1, 2, 3, \ldots ; \quad N_1 = M(\text{Max } A) ; \end{cases}$$

$N_2, N_4, N_8, \ldots, N_2 j \ldots$, jusqu'à stabilisation. Max A^* est l'ensemble des éléments de la matrice stabilisée.

Max est une opération qui d'une part sélectionne les éléments maximaux à l'intérieur de chaque case matricielle (puisque des éléments non nuls d'indices différents sont incomparables) et qui d'autre part supprime l'élément nul 0 ; $\cup$ opère sur les cases de mêmes indices.

On peut utiliser des index dans chaque case pour tenir compte des considérations de 4-II-2.

b. Algorithmes matriciels à pas.

Dans F1 et M1 aucune comparaison d'éléments n'est faite tant qu'il est possible d'effectuer de nouvelles combinaisons ; il y a par contre dans F2

une interpénétration maximale du calcul et de l'insertion des nouvelles combinaisons, interpénétration qui va être utilisée par les algorithmes à pas. Ils vont donc opérer grossièrement de la façon suivante : on part du sous-ensemble Max A de A écrit matriciellement. On emploie alternativement les deux règles suivantes :

1. on choisit exclusivement un couple (α,β) et (β,γ) de cases ou un couple ligne-colonne (α,i)-(i,β) ou un couple case-ligne (α,β)-(β,i) ou un couple colonne-case (i,α)-(α,β), et on recherche les $*$ - composés correspondants ;

2. dès qu'ils sont formés, on les ajoute à leurs justes cases dans la matrice s'ils ne sont pas inférieurs ou égaux à des éléments de la matrice, et on supprime les éléments qui leur sont inférieurs ;

jusqu'à la stabilisation du procédé. Max A^* est l'ensemble des éléments de la matrice obtenue.

Donnons maintenant deux options possibles pour la progression des indices α et β dans (1).

1. Donnons d'abord une option assez naturelle, inspirée de M1 : à chaque itération ou tour, on recherche les $*$ - composés des éléments de toutes les façons possibles (en tenant compte bien sûr des conventions de 4-I-3 et de celle en tête du chapitre).

Algorithme M2/ou algorithme matriciel naturel.

On part du sous-ensemble Max A de A écrit matriciellement. On itère jusqu'à stabilisation, les deux règles suivantes :

1. on recherche les $*$ - composés des I^2 couples $\alpha^{ème}$ ligne - $\beta^{ème}$ colonne ;

2. dès que pour un tel couple, les $*$ - composés sont formés, on les ajoute à leurs justes cases dans la matrice s'ils ne sont pas inférieurs ou égaux à des éléments de la matrice, et on supprime les éléments qui leur sont inférieurs ;

$\text{Max } A^*$ est l'ensemble des éléments de la matrice ainsi obtenue.

Le nombre d'opérations à chaque itération au niveau des cases matricielles est I^3, comme pour l'algorithme M1. M2 converge cependant après un nombre d'itérations moindre que M1, M2 utilisant à chaque pas toutes les informations connues tandis que M1 attend la fin de chaque tour pour les utiliser.

La rapidité de la convergence de M2 dépend de l'ordre du choix des I^2 couples $\alpha^{\text{ème}}$ ligne - $\beta^{\text{ème}}$ colonne, alors que la convergence de M1 en est indépendante (voir [TOMESCU 2] p. 52-55 pour un choix très efficace des couples (α, β) valable en particulier quand G est un monoïde avec une loi d'absorption).

2. Donnons maintenant une variante assez particulière puisque nécessitant les hypothèses supplémentaires suivantes : deux éléments non nuls de premiers indices différents sont incomparables, $*$ est pseudo-associatif et le premier indice associé aux $*$ - composés de deux éléments $b_{\alpha i}$ et $c_{i\beta}$ est supérieur ou égal à α (c'est-à-dire ils sont de la forme $a_{\geq \alpha, \gamma}$; $\geq$ est la relation d'ordre des entiers naturels) (conditions E).

Algorithme M2' ou algorithme matriciel différentié.

On part du sous-ensemble Max A de A écrit matriciellement. Pour $\alpha = 1$, puis 2... enfin I, on itère jusqu'à stabilisation les deux règles suivantes :

1. On recherche les $*$ - composés de la $\alpha^{\text{ème}}$ ligne et des I différentes colonnes (on effectue le produit matriciel $\alpha^{\text{ème}}$ ligne $*$ $\beta^{\text{ème}}$ colonne, pour $\beta = 1, 2, \ldots, I$) ;

2. Dès que pour un tel produit $\alpha^{ème}$ ligne $\overset{U}{*}$ $\beta^{ème}$ colonne les $*$-composés sont formés, on les ajoute à leurs justes cases dans la matrice s'ils ne sont pas inférieurs ou égaux à des éléments de la $\alpha^{ème}$ ligne ou/ou s'ils ne sont pas nuls, et on supprime les éléments qui leur sont inférieurs ;

Max A^* est l'ensemble des éléments de la matrice stationnaire ; on obtient successivement, à chaque stabilisation, les éléments de Max A^* dont le premier indice est 1, puis 2, ..., enfin I.

c. Remarques.

1. Les interpénétrations, dans M1 d'une part, dans M2 et M2' d'autre part, du calcul (1) et de l'inclusion dans la matrice (2) des $*$-composés sont extrêmes, à l'exemple de celles des algorithmes F1 d'une part, F2 d'autre part ; des interpénétratinns intermédiaires peuvent évidemment être imaginées ; on peut aussi par exemple donner à l'algorithme M2' une forme à tours, où on ajoute dans leurs justes cases les $*$-composés seulement après avoir fait les I produits $\alpha^{ème}$ ligne $\overset{U}{*}$ $\beta^{ème}$ colonne, avec $\beta = 1, 2, \ldots, I$.

2. La considération des couples ligne-colonne au lieu de couples case-ligne ou colonne-case dans les trois algorithmes matriciels précédents est une pure affaire de convention.

3. Corrélation entre les conditions M, D et E. Les conditions I et M laissent présumer que deux éléments non nuls d'indices différents sont incomparables : en effet $b_{\alpha\beta} \leq c_{\gamma\delta}$ entraîne :

si $\beta \neq \delta$, pour tout élément de la forme $d_{\beta\varepsilon}$: $b_{\alpha\beta} * d_{\beta\varepsilon} = 0$

si $\alpha \neq \gamma$, pour tout élément de la forme $d_{\varepsilon\alpha}$: $d_{\varepsilon\alpha} * b_{\alpha\beta} = 0$.

La pseudo-associativité (condition E) entraîne, aussi, souvent que les $*$ - composés de deux éléments $b_{\alpha i}$ et $c_{i\beta}$ sont de la forme $d_{\alpha\beta}$.

4. Enfin, l'algorithme récursif F3 peut évidemment donner naissance à des algorithmes matriciels récursifs, dont les matrices successives sont emboîtées les unes par rapport aux autres ; à chacune d'elles, on applique l'un des algorithmes matriciels précédents. Nous verrons une application dans l'exemple II-2c.

I.2. Algorithme de McNAUGHTON-YAMADA-ROY-WARSHALL.

Si les conditions C et d'existence d'un support matriciel sont toutes deux vérifiées, on peut mélanger les algorithme C et matriciels.

Cependant, l'intérêt de M2' étant de fournir le plus rapidement possible les éléments définitifs de la première ligne, ensuite ceux de la deuxième ligne, ..., il faut mieux utiliser l'algorithme C lors de chaque itération de l'algorithme M2' qu'inversement appliquer l'algorithme M2' à chacune des phases de l'algorithme C.

Au lieu d'énoncer les algorithmes CM1, CM1', CM2, M2'C et CM3, donnons un algorithme matriciel C valable, quand deux éléments non nuls d'indices différents sont incomparables, quand les $*$ - composés de deux éléments $b_{\alpha i}$ et $c_{i\beta}$ ont pour indices associés (α,β) et quand les opérations $*$ sont pseudo-associatives et isotones (les conditions I, C, M, D et E sont donc vérifiées mais non la condition de finitude B3) :

Algorithme CM ou algorithme de McNAUGHTON-YAMADA-ROY-WARSHALL

On part du sous-ensemble Max A de A écrit matriciellement.

Pour i = 1, 2,...,I, on itère le processus (1,2,3) suivant :

1. Si la case (i, i) a des éléments, on les remplace par

Max {éléments de la case (i, i)}*

c'est-à-dire les éléments constructibles à partir des éléments de la case (i,i) qui sont maximaux.

2. Si la case (i, i) n'est pas vide, on effectue pour β= 1,2,..., i-1, i+1, ...,I, et pour les opérations $*$, les combinaisons

{éléments de la case (i,i)}$*${éléments de la case (i,β)} ;

on les place après réduction dans la case (i,β) s'ils ne sont pas inférieurs ou égaux à des éléments de cette case ou s'ils ne sont pas nuls, et on élimine les éléments de cette case qui leur sont inférieurs.

3. On effectue, pour α, β = 1, 2, ...,I avec $\alpha \neq$ i et pour les $*$, les combinaisons :

{éléments de la case (α,i)}$*${éléments de la case (i,β)} ;

on les place après réduction dans la case (α, β) s'ils ne sont pas inférieurs ou égaux à des éléments de cette case ou s'ils ne sont pas nuls, et on élimine les éléments de cette case qui leur sont inférieurs.

Max A^* est l'ensemble des éléments de la matrice obtenue.

Le parallèle impliqué par le nom propre de CM entre les algorithmes de [McNAUGHTON-YAMADA] (voir II-1 c, III-2b) et de [ROY 1] ou [WARSHALL] (voir II-2b, III-3c) n'est pas nouveau : [VEILLON] montre que le deuxième algorithme (celui de ROY-WARSHALL) est un cas particulier du premier.
CM est beaucoup plus général que ces deux algorithmes.

Démonstration .

Décomposons la famille $*$ d'opérations en I familles $*_i$ d'opérations, restrictions de $*$ aux couples d'éléments $b_{\alpha\beta}$ et $c_{\gamma\delta}$ tels que $\beta = \gamma = i$; les opérations $*_i$ sont pseudo-associatives.

La première itération de l'algorithme fournit les $*_1$ - composés maximaux. Supposons que la $(i-1)^{ème}$ itération fournisse les $*_{<i}$ - composés maximaux et montrons que la $i^{ème}$ donne les $*_{\leq i}$ - composés maximaux.

En effet, ils sont de la forme :

$$\underbrace{b_{\alpha\beta} *_\beta c_{\beta\gamma} *_\gamma \ldots *_\varphi \ell_{\varphi i}}_{p} *_i \underbrace{f_{i\psi} *_\psi \ldots *_\eta g_{\eta i}}_{q} *_i \underbrace{h_{i\zeta} *_\zeta \ldots *_\zeta \ell_{\zeta i}}_{r} *_i \underbrace{m_{i\rho} *_\delta \ldots *_\delta d_{\delta\varepsilon}}_{s \qquad t}$$

où les accolades désignent des $*_{<i}$ - composés. L'étape 1 permet de construire l'élément $u = q *_i r *_i \ldots *_i s$ à partir des $*_{<i}$ - composés ou un élément supérieur, l'étape 2 permet de construire un élément supérieur ou égal à $p *_i u$ et l'étape 3 $(p *_i u) *_i t$.

Remarque 1.

Cet algorithme est convergent si les Max {éléments de la case (i,i)}* vérifient la condition de finitude B3. Sinon ce n'est pas un algorithme à proprement parler ; il a alors l'intérêt de représenter Max A^* à l'aide des {Max {éléments de la case (i,i)}* , $i = 1, 2, \ldots, I$}, de décomposer le problème

de la recherche d'éléments maximaux en des problèmes de même nature (existence d'une infinité d'éléments maximaux ; cette infinité est représentée symboliquement pour pouvoir poursuivre l'application de l'algorithme) mais plus reinstreints.

Remarque 2.

CM forme les éléments de Max A^{*} sans redondance si, dans les phases 1, les éléments maximaux sont formés sans redondance ; autrement dit le couple d'une suite d'éléments de A et d'une suite adéquate d'opérateurs $*$ est utilisé au plus une fois pour fournir l'$*$ - composé correspondant.

Cela se montre par induction : supposons que les combinaisons p, q, r, ..., s, t sont effectuées une seule fois ; comme $u = q *_i r *_i \ldots *_i s$ est supposé aussi formé une seule fois, $(p *_i u) *_i t$ est formé une seule fois.

Remarque 3.

CM peut être déguisé en un algorithme en file, qui sera appelé CF ; les éléments de mêmes indices sont traités simultanément ; à la $i^{ème}$ itération, sont traités d'abord les éléments d'indices (i, i), puis sont composés les éléments d'indices (α, i) et (i, i), enfin ceux d'indices (α, i) et (i, β) .

Remarque 4.

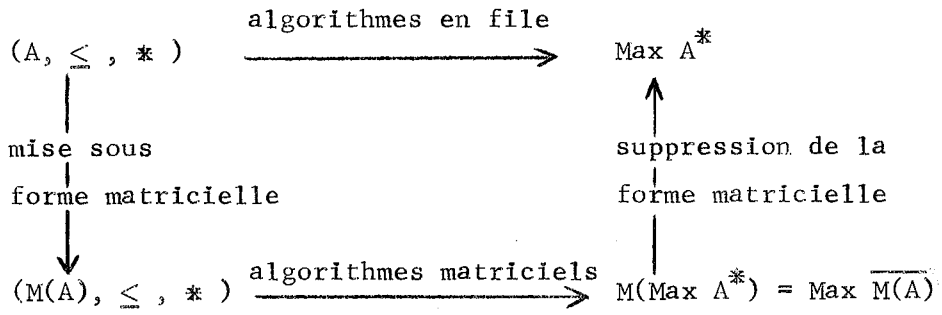
CM, CM1 et CM2 coïncident lorsque les éléments des cases (i, i) ne sont pas combinés.

Les algorithmes CM et CF présentent l'avantage sur CM1, CM2, CF1 et CF2, de reinstreindre au cours de la $i^{ème}$ itération la recherche la plus générale d'éléments maximaux (chapitre 4) aux seuls éléments de la case (i,i) pendant l'étape 1 ; les étapes 2 et 3 se réduisent à des développements

(voir le chapitre 7). Dans les étapes 2 et 3, il n'y a pas lieu de différencier processus par pas et processus par tours à cause de la remarque b de 4-II-3 ; par contre dans l'étape 1, on peut employer l'un ou l'autre.

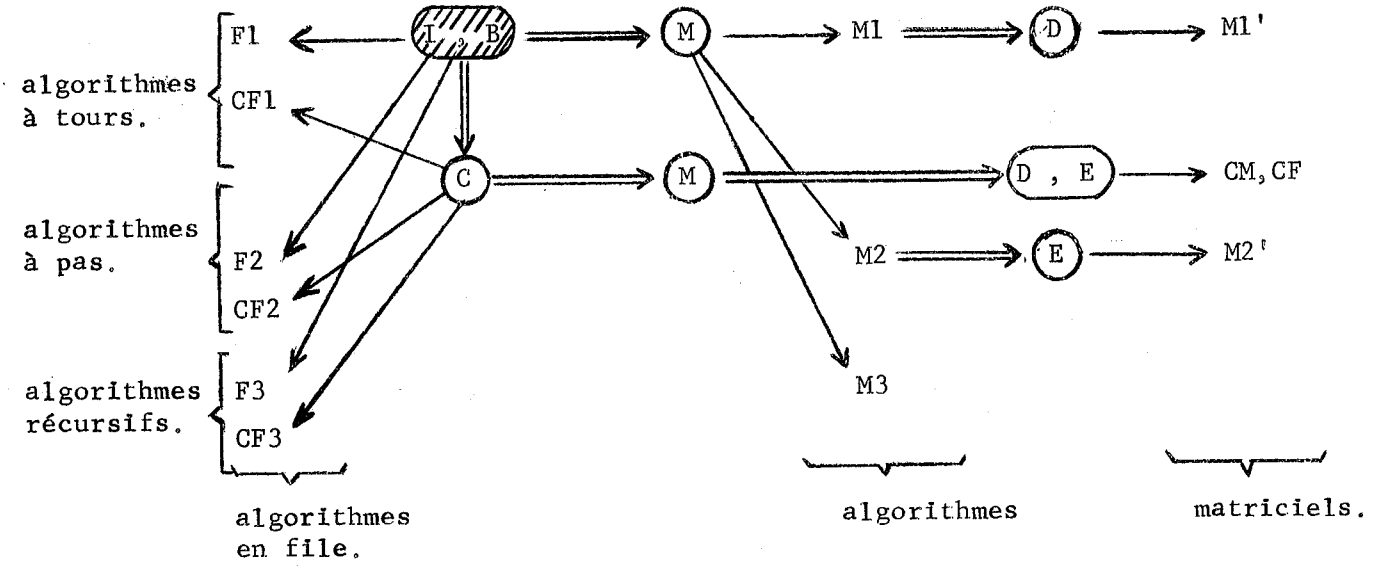
I.3 Conclusion.

La phase opératoire des algorithmes précédents est donc le calcul sur matrice, leur phase préparatoire étant la mise sous forme matricielle (qui est immédiate) de Max A et leur phase finale étant la désintégration (immédiate) matricielle. Ce processus peut être ainsi schématisé, si M(A) désigne l'écriture matricielle de A et si la condition D, en plus des conditions I, B et M, est vérifiée :



Dans le cas où la condition B3 n'est pas vérifiée, les algorithmes n'existent pas pour calculer Max A^*, mais l'algorithme matriciel CM permet de décomposer son calcul en des calculs moins généraux.

Le schéma suivant récapitule les principaux algorithmes rencontrés jusqu'à maintenant (chapitre 4 et 5) et qui seront utilisés dans la suite :


F1
CF1
algorithmes à tours.
algorithmes à pas.
F2
CF2
algorithmes récursifs.
F3
CF3
A, B
C
M
M
M1
M2
M3
D
D , E
E
M1'
CM, CF
M2'
algorithmes en file.
algorithmes
matriciels.

II - APPLICATION A L'ENUMERATION DES ELEMENTS MAXIMAUX DES MONOIDES ISOTONES

II.1. L'énumération des éléments maximaux des monoïdes isotones.

Soit un monoïde ordonné G à opération (appelée produit et notée .) isotone, A une de ses parties finies, et soit A° le sous-monoïde de G engendré par A. O désignera le plus petit élément et l'élément nul de G.

a. Génération des éléments maximaux d'un monoïde isotone satisfaisant à la condition de finitude B.

Les éléments maximaux de A° étant en nombre fini et pouvant être obtenus à partir de A en un nombre fini de produits, les algorithmes en file permettent de les déterminer.

Des exemples de monoïdes isotones sont les monoïdes finis (monoïdes ayant un nombre fini d'éléments) à opération isotone et en particulier les monoïdes finis non ordonnés : en effet ils vérifient trivialement la condition I puisque, mis à part O, les éléments de G sont deux à deux incomparables et puisque O a pour produit avec tout élément de G, O ; ils vérifient aussi la condition B. Des monoïdes finis non ordonnés particuliers sont les monoïdes de carré nul [BENZAKEN 1] p. 180, les monoïdes de puissances n_a nulles (monoïdes possédant un élément nul O et tels que , pour tout élément a distinct de l'élément neutre, il existe un entier fini n_a tel que, quels que soient les n_a-1 mots P, Q, ..., R de G , aPaQa...aRa = O), les monoïdes de longueur p nulle (monoïdes possédant un élément nul et dont les mots ayant pour longueur p, ou un entier supérieur à p, sont nuls).

b. Génération des éléments maximaux d'un monoïde isotone satisfaisant aux conditions M, D et E.

Si $A^\bullet$ vérifie les conditions M, D et E, c'est-à-dire :

- si, à tout élément non nul $b_{\alpha\beta}$ de G est attaché un couple ordonné d'indices (α, β) (avec $\alpha, \beta = 1, 2, \ldots, I$) ,
- si, le produit de deux éléments $b_{\alpha\beta}$ et $c_{\gamma\delta}$ peut-être non nul seulement si $\beta = \gamma$ et si, non nul, il a pour indices associés (α, δ) ,
- et si les éléments non nuls d'indices différents sont incomparables,

alors la condition C est aussi vérifiée vis-à-vis des I opérations $\ldots_i$

restrictions de $\ldots_i$ aux couples d'éléments $b_{\alpha\beta}$ et $c_{\gamma\delta}$

tels que $\beta = \gamma = i$; on peut appliquer l'algorithme matriciel CM , et , si $A^\bullet$ vérifie la condition de finitude B3, tous les algorithmes matriciels précédents.

c. Exemples.

Des exemples de tels monoïdes isotones sont les monoïdes A° de chemins dans les graphes finis engendrés par l'ensemble A de leurs arcs :

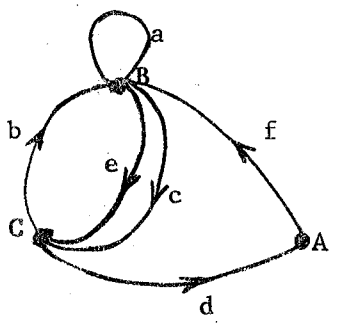
- deux chemins sont incomparables ;
- le produit est l'opération interne non commutative faisant correspondre à deux chemins $x_k \ldots x_\ell$ et $x_m \ldots x_n$ de A° (les sommets du graphe sont $x_1, x_2, \ldots, x_I$), le chemin concaténation $x_k \ldots x_\ell \ldots x_n$ si $\ell = m$, sinon l'élément nul . $\ldots_i$ désigne l'opération interne non commutative faisant correspondre à deux chemins $x_k \ldots x_\ell$ et $x_m \ldots x_n$ le chemin $x_k \ldots x_i \ldots x_n$ si $\ell = m = i$ $(i = 1, 2, \ldots, I)$, sinon l'élément nul.

Les monoïdes de chemins vérifiant la condition B3 seront étudiés en II.2. Parmi les monoïdes infinis de chemins, parlons tous de suite du monoïde de tous les chemins d'un graphe.

CM est alors l'algorithme de [McNAUGHTON, YAMADA] donnant les expressions régulières d'un graphe, dans le cas où les informations portées par les arcs sont les arcs eux-mêmes. Max {éléments de la case (i,i)}(i), ou l'ensemble des chemins d'origine et d'extrémité le $i^{ème}$ sommet du graphe et de sommets intermédiaires des sommets d'indices au plus i, peut être représenté à l'aide de l'opérateur étoile (noté * : $a^* = 1 \cup a \cup a^2 \cup \ldots = 1 \cup a^+$, si 1 est l'élément unité ; a^+ est le monoïde engendré par a à l'aide de .) des expressions régulières.

Les opérateurs * et ^+_i représentant sans répétition les chemins constructibles à partir des chemins de la case (i, i) à l'aide de . , la remarque 2 de I.2 implique que les chemins trouvés par CM le sont sans répétition.

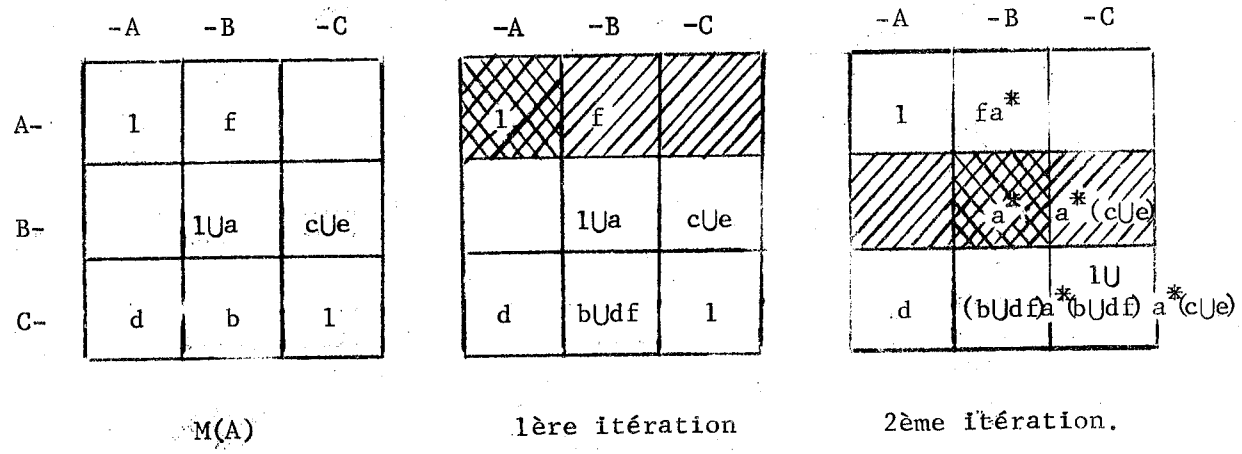
Donnons un exemple de recherche des expressions régulières d'arcs d'un graphe.

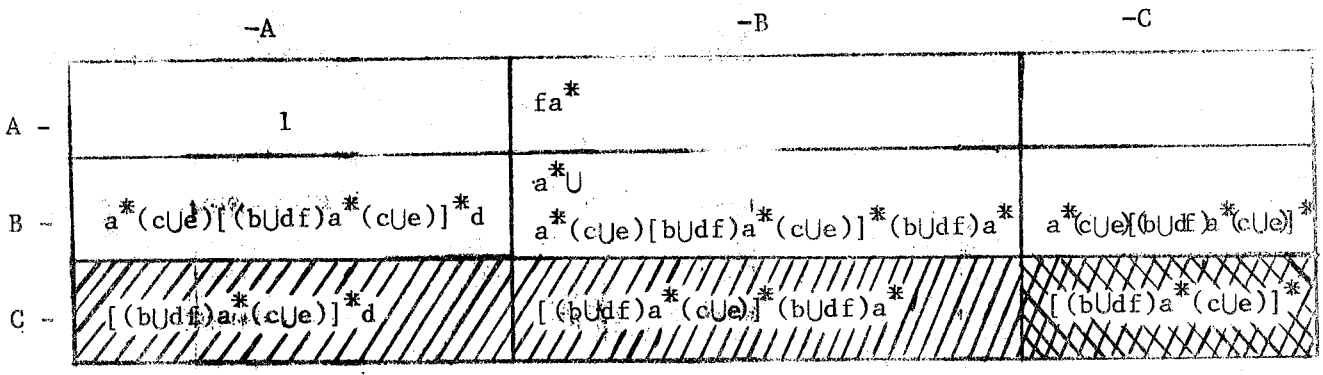


Pour simplifier les notations, nous posons :

a = boucle BB
b = arc CB
c = arc BcC
d = arc CA

CM appliqué aux arcs mis sous forme matricielle donne :





3ème itération.

étape 1
étape 2
étape 3
} de la ième itération.

Les expressions régulières sont obtenues de façon unique, mais leurs écritures sont souvent compliquées ; c'est ainsi que $a^*Ua^*(cUe)[(bUdf)a^*(cUe)]^*(bUdf)a^*$ admet pour forme simplifiée : $a^*[(cUe)(bUdf)a^*]^*$.

II.2. Application à l'énumération de chemins dans les graphes finis.

Un travail de synthèse considérable a déjà été fait dans [PAIR, DERNIAME], où il est donné des algorithmes de recherche de chemins et leurs transformés par des homomorphismes. Rappelons que ces homomorphismes associent à un ensemble de chemins joignant un point d'un graphe à un autre, une information qui

- prend l'une des valeurs vrai ou faux dans le problème d'existence de chemins,
- ou est un réel dans le problème de plus courte ou de plus longue distance,
- ou est un chemin dans le problème de plus courts ou de plus longs chemins,
- ou est un entier dans le problème de nombres de chemins,
- ou est une probabilité dans le problème de probabilité de passage d'un point à un autre.

On peut voir aussi [KAUFMANN 2] p. 326 qui met en relief la structure de monoïde.

L'étude des éléments maximaux permet une synthèse ayant de nouvelles dimensions : d'une part les algorithmes de recherche de chemins et leurs homomorphes apparaissent comme des applications des algorithmes en file et matriciels

vus précédemment (leur classement en particulier sera facilité) ; d'autre part ils sont plus généraux (par exemple, la généralisation de l'algorithme de ROY-WARSHALL obtenue ici ne se limite pas à l'énumération des seuls chemins et circuits élémentaires).

a. Généralités.

Prenons pour monoïdes finis à élément nul (le chemin vide) engendrés par A, A° :

- un monoïde de chemins de longueur supérieure à p nuls si on recherche les chemins de longueur p ou au plus égale à p,
- un monoïde de puissances n_a nulles si on recherche les chemins passant moins de n fois par l'arc a, et en particulier un monoïde de carré nul si on recherche les chemins ou circuits simples, les chemins ou circuits eulériens,
- un monoïde de puissances n_i nulles (monoïde dont les éléments contenant n_i fois la lettre x_i sont nuls, $i = 1, 2, \ldots, I$) si on recherche les chemins ou circuits passant moins de n_i fois par le sommet x_i ou exactement $n_i - 1$ fois, et en particulier de carré des sommets nul si on recherche les chemins élémentaires ou seulement les chemins hamiltoniens,
- un monoïde satisfaisant à plusieurs de ces propriétés ou, plus généralement si on recherche les éléments $\{a^j\}$, un monoïde isotone satisfaisant la condition B et admettant $\{a^j\}$ pour partie.

En attachant à tout chemin (non nul) le couple ordonné de son sommet origine et de son sommet extrémité, A° vérifie les conditions I, B, C, M, D et E et on peut utiliser tous les algorithmes donnés en file et matriciels, avec les remarques simplificatrices évidentes suivantes :

- puisque la condition D est vérifiée, M1' est applicable et M1 n'a plus d'intérêt ;
- Max A = A et les clauses relatives à la suppression d'éléments inférieurs n'ont plus de raison d'être, puisque les éléments sont incomparables.

b. Parlons des algorithmes suivants pour leurs particularités ou parce que certaines de leurs applications sont déjà connues :

L'algorithme M1' devient pratiquement celui de [KAUFMANN 1] p. 313-336.

Une application de l'algorithme CM est l'algorithme de WARSHALL de [PAIR, DERNIAME] p.22 (les éléments dans les cases (α, α) ne sont pas combinés).

Les algorithmes CF1 et CF2 sont identiques à cause de la remarque b de 4-II-3.

Remarque 1.

Un algorithme de recherche d'éléments maximaux doit évidemment les donner sans omission, mais aussi si possible sans répétition ; en effet, l'absence de redondance a deux avantages :

- elle évite la recherche de réductions possibles ;
- elle peut-être une mesure indirecte de l'efficacité d'un algorithme, toutes ses opérations, quand elles sont définies, étant nécessaires puisqu'elles donnent chacune naissance à un nouvel élément.

Dans l'exemple qui suit relatif à la recherche des chemins élémentaires d'un graphe, nous verrons que les seuls algorithmes dont on a parlé jusqu'à maintenant sans répétition sont CF1 ou CF2 , et CM (des couples d'éléments donnant un même élément seront encadrés dans l'exemple qui suit).

Plus généralement, la remarque 2 de I.2 entraîne que l'algorithme CM appliqué à la recherche de chemins dans les graphes les donne toujours sans répétition, à condition que ceux formés au cours des phases 1 le soient sans répétition. En effet, deux chemins non vides sont incomparables et un chemin non vide est le produit d'une seule suite d'éléments de A .

Montrons qu'il est possible de modifier les opérations de tous les algorithmes, de façon à les rendre irredondants :

- pour les algorithmes C, $1'_i$. - composé de deux chemins

$x_k \ldots x_\ell$ et $x_m \ldots x_n$ de G sera défini si et seulement si

$$\begin{cases} \ell = m = i, \\ \text{nombre de fois où } x_k \ldots x_\ell \text{ passe par } x_i = \text{nombre de fois où} \\ \quad x_m \ldots x_n \text{ passe par } x_i, \text{ augmenté d'un éventuellement,} \\ x_k \ldots x_i \ldots x_n \text{ n'est pas l'élément nul de } G \text{ ;} \end{cases}$$

- pour les autres algorithmes, 1'. - composé de deux chemins

$x_k \ldots x_\ell$ et $x_m \ldots x_n$ de G sera défini si et seulement si

$$\begin{cases} \ell = m, \\ \text{nombre de sommets de } x_k \ldots x_\ell = \text{nombre de sommets de} \\ x_m \ldots x_n, \text{ augmenté éventuellement d'un,} \\ x_k \ldots x_\ell \ldots x_n \text{ n'est pas l'élément nul de } G \text{ .} \end{cases}$$

D'autres définitions du produit sont possibles pour éviter les répétitions.

Remarque 2.

L'obtention des chemins de longueur p, des chemins élémentaires de longueur p, des chemins passant n_i fois par x_i (i = 1, 2, ...I), des chemins passant n_a fois par l'arc a se fait par une séléction des chemins de longueur inférieure ou égale à p , des chemins contenant au plus n_i fois le sommet x_i , des chemins passant n_a fois au plus par l'arc a respectivemet. Nous verrons plus loin une autre méthode pour trouver ces chemins (7-II-1).

D'autres problèmes sont directement liés à ceux-là : énumération des circuits élémentaires de longueur i = 2, 3, ..., I (ou circuits hamiltoniens) comme produits de chemins élémentaires de p et i-p arcs (i = 2,3...,I), les sommets origine du premier arc et extrémité du ième étant identiques (p est fixé, compris entre 1 et i-1) ; énumération des facteurs d'un graphe (on recherche d'abord tous les circuits élémentaires, puis les couvertures des sommets par ces circuits élémentaires par élimination) ; énumération des dissections d'un graphe (en calculant les chemins et les circuits élémentaires et en les associant). Voir [BENZAKEN 1], [KAUFMANN 2] p. 316-346.

c. Exemple.

Prenons celui de la page 307 de [KAUFMANN 1] :

A = AB ∪ AC ∪ AE ∪ BC ∪ BE ∪ CD ∪ DC ∪ DE ∪ EA ∪ ED

Recherchons les chemins élémentaires du graphe dont l'ensemble des arcs est A. Nous soulignerons les chemins hamiltoniens.

1. Appliquons l'algorithme M1' à l écriture matricielle M(A) de A :

$M(A) = N_1 =$

	AB	AC		AE	A-
		BC		BE	B-
			CD		C-
		DC		DE	D-
EA			ED		E-
-A	-B	-C	-D	-E	

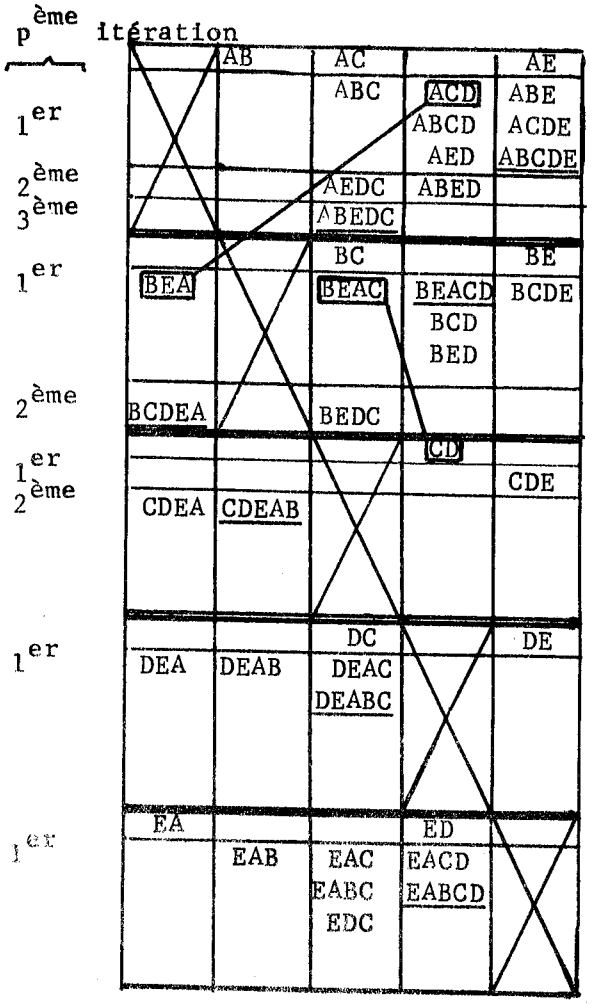
$N_2 =$

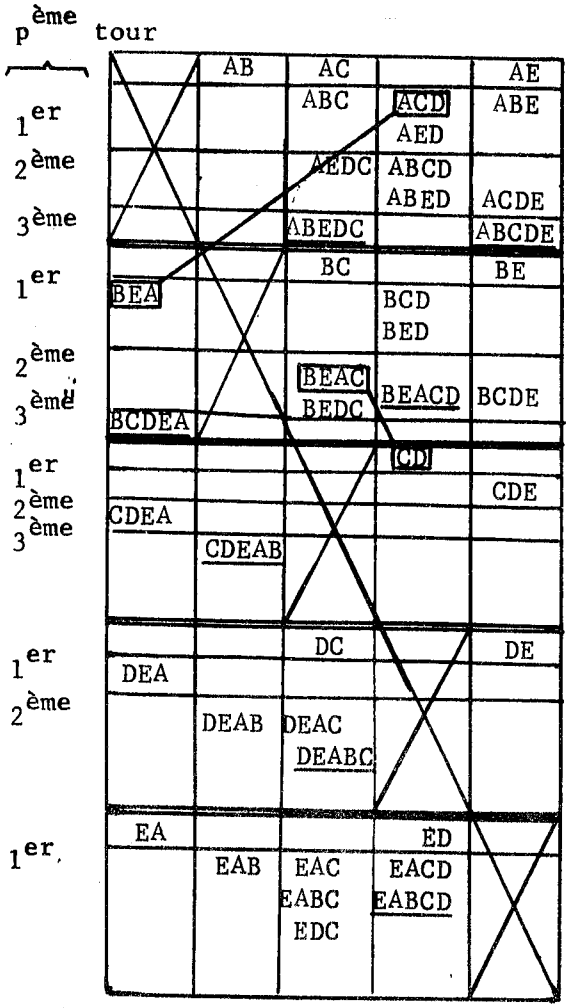
	AB	AC ABC	ACD AED	AE ABE
BEA		BC	BCD BED	BE
			CD	CDE
DEA		DC		DE
EA	EAB	EAC EDC	ED	

$N_4 =$

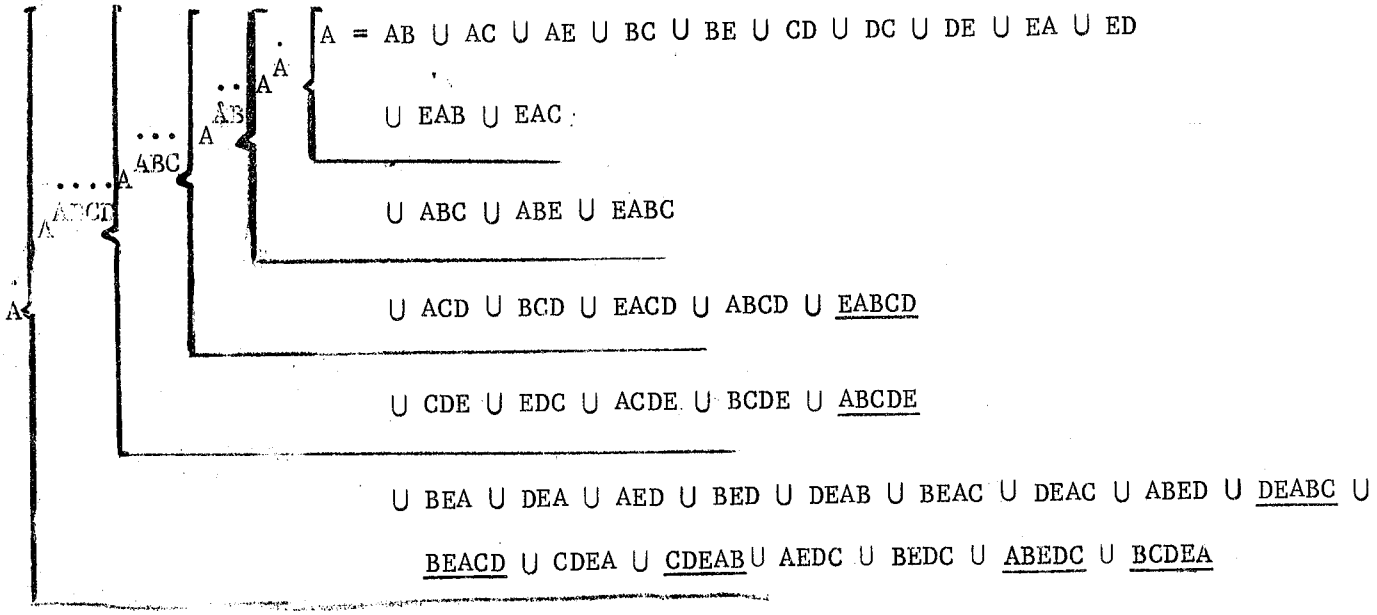
	AB	AC ABC AEDC ABEDC	ACD AED ABCD ABED	AE ABE ACDE ABCDE
BEA BCDEA		BC BEAC BEDC	BCD BED BEACD	BE BCDE
CDEA	CDEAB		CD	CDE
DEA	DEAB	DC DEAC DEABC		DE
EA	EAB	EAC EDC EABC	ED EACD EABCD	

2. L'algorithme matriciel différentié M2' (qui est à pas) et l'algorithme correspondant à tours (les adjonctions des produits ne sont faites qu'après le calcul des composés d'une $\alpha^{ème}$ ligne et des I colonnes) donnent respectivement :





3. Utilisons les algorithmes CF1 ou CF2, avec l'ordre A, B, C, D, E.



On peut faire les remarques simplificatrices suivantes :

- deux chemins élémentaires apparus au cours de la même itération ne peuvent pas avoir de produit, puisqu'ils contiennent tous deux une même lettre intérieure;
- deux chemins élémentaires dont le nombre total des sommets est supérieur à I ne peuvent pas avoir de produit ; en particulier un chemin hamiltonien ne peut pas avoir de produit avec un autre chemin ;
- un chemin élémentaire apparu au cours d'une itération i ne peut avoir d'$*_i$ - composé au cours de cette itération ; il en résulte en particulier que, si on ne recherche que les chemins hamiltoniens, il est inutile à la

dernière itération de faire apparaître des chemins élémentaires non hamiltoniens ;

- à chaque itération, on peut, nonobstant la convention c de 4-I-3, grouper les chemins ayant même origine et même extrémité ; on aboutit alors à l'algorithme matriciel CF.

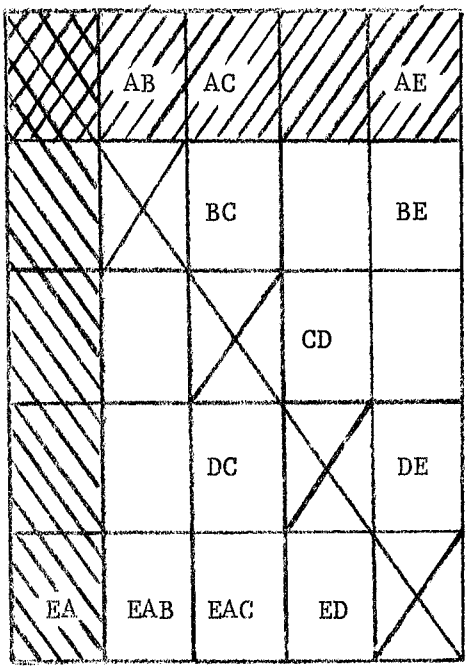
4. Dans le cas de la recherche des seuls chemins élémentaires, la $i^{ème}$ itération de l'algorithme CM présente les caractéristiques suivantes :

les cases de la diagonale principale restent vides comme celles de M(A) ;

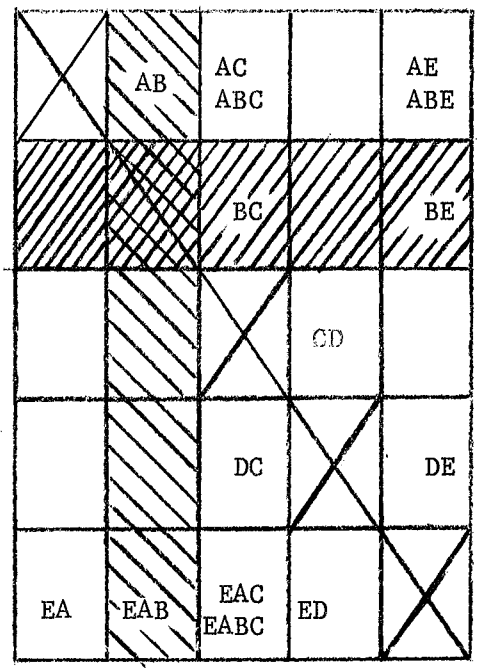
les éléments des $i^{èmes}$ ligne et colonne restent invariants ;

CM n'a pas de phase 1 .

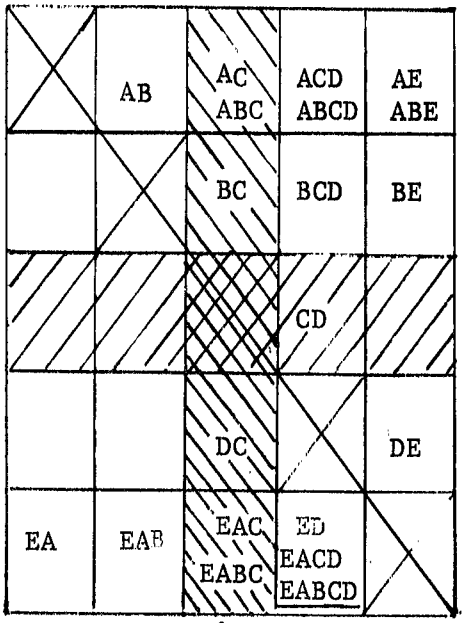
Appliquons ces remarques à l'exemple étudié, en prenant pour les itérations l'ordre A, B, C, D, E :



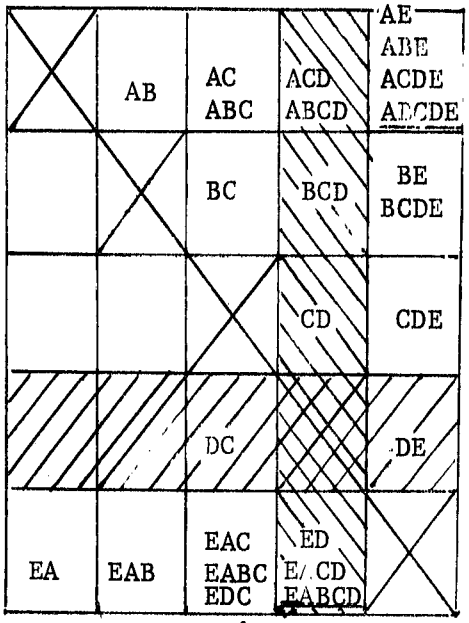
$M_1 = M(A) \cup (M(A) \overset{\circ}{A} M(A))$



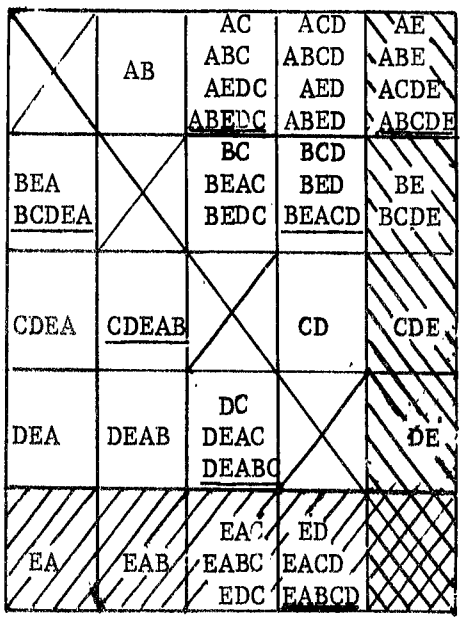
$M_2 = M_1 \cup (M_1 \overset{\circ}{B} M_1)$



$M_3 = M_2 \cup (M_2 \overset{G}{\cdot} M_2)$



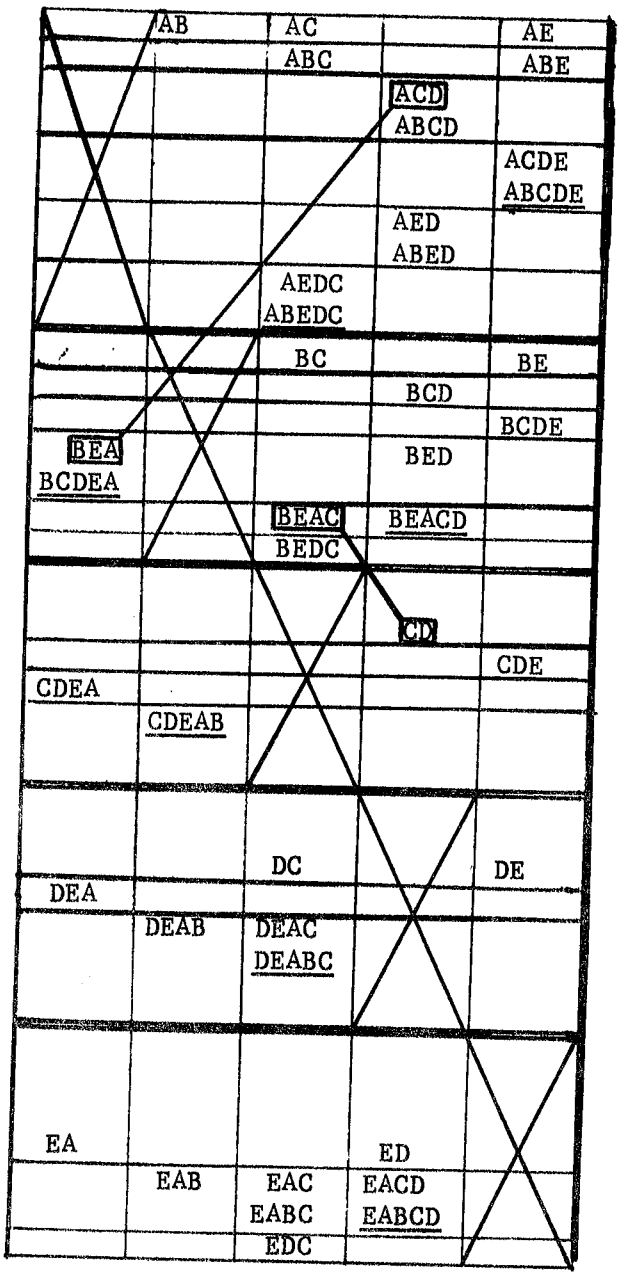
$M_4 = M_3 \cup (M_3 \overset{D}{\cdot} M_3)$



$M_5 = M_4 \cup (M_4 \overset{E}{\cdot} M_4)$

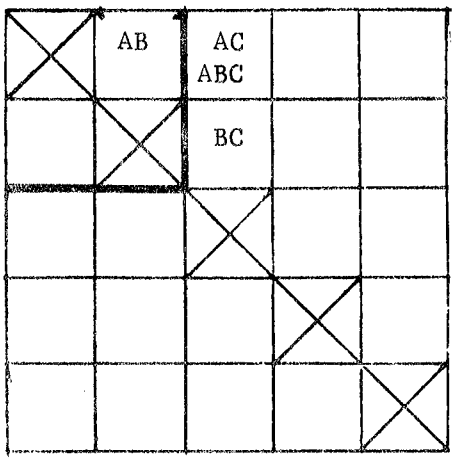
Comme il est indiqué dans [PAIR, DERNIAME], le nombre d'opérations $\cup$ et le nombre de produits est chacun de $I\ (I-1)\ (I-2)$.

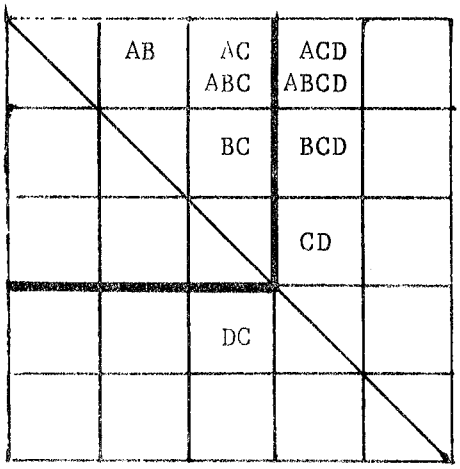
5. Appliquons maintenant l'algorithme M2'C, avec l'ordre A, B, C, D, E .

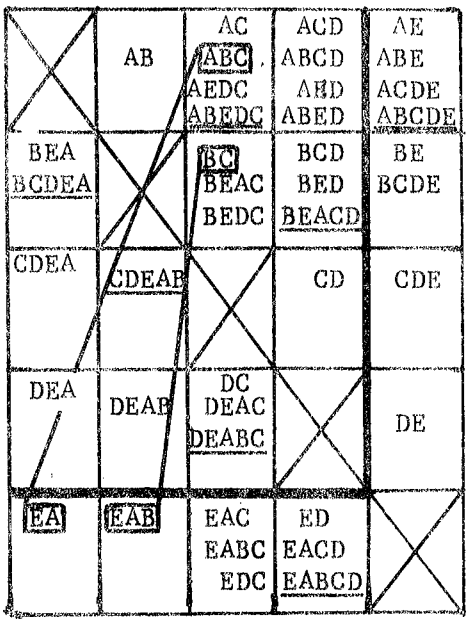


Pour éviter les répétitions ( ), il suffit par exemple d'imposer de ne multiplier qu'à droite et que par des arcs.

6. Appliquons maintenant l'algorithme récursif défini de la façon suivante : les ensembles d'arcs sur lesquels on opère sont successivement ceux des sous-graphes de sommets {A, B}, {A, B, C} , {A, B, C, D} , {A, B, C, D, E} ; les chemins élémentaires de chaque sous-graphe sont obtenus à l'aide de CM1' ou CM2 , avec l'ordre A, B, C, D, E.







On peut éviter les répétitions en multipliant à chaque itération les chemins élémentaires obtenus à la fin de l'itération précédente par les arcs nouvellement introduits, puis en recherchant les produits par rapport à la variable nouvellement introduite : c'est l'algorithme de DANTZIG de [PAIR, DERNIAME] p. 25.

d. Autre triplet pour énumérer des chemins de graphes sans circuit.

Prenons les mêmes ensemble de départ et relation d'ordre que précédemment, mais comme opérations autant d'opérations unaires $*_a$, $*_b$, $*_c$,... que d'arcs a, b, c,... faisant correspondre à un chemin μ dont l'extrémité est respectivement l'origine de a, b, c,... le chemin concaténation de μ et respectivement a, b, c, ..., sinon l'élément nul. Si le graphe est sans circuit (ce qui est équivalent à : il existe un ordre total des arcs $\ldots \leqslant_T a \leqslant_T \ldots \leqslant_T b \leqslant_T \ldots$ tel que si a et b sont deux arcs quelconques, ba est un chemin entraîne $a \leqslant_T b$), alors la condition C est vérifiée puisque son premier membre est nul et l'ensemble des chemins est égal à

$$\left[\{ (A \cdots) a^{*\ldots} \right] b^{*\ldots} \}^{\cdots}$$

C'est l'algorithme de [PAIR, DERNIAME] p. 47, Son efficacité résulte d'une part de l'unarité des opérations choisies (chaque $*_a$ est applicable à I cases au lieu de I^2 cases pour un opérateur binaire), d'autre part de l'emploi d'une seule occurrence de chacune de ces opérations (rendu possible par l'hypothèse d'un graphe sans circuit).

III - APPLICATION A LA FERMETURE TRANSITIVE DE MATRICES

III-1. Pseudo-fermeture transitive d'une matrice définie sur l'ensemble des parties d'un ensemble à opérations isotones.

a. Définitions.

Appelons puissance deuxième $\overset{\cup}{*} M^2$ ou M^2 d'une matrice carrée $M = (m_{(\alpha\beta)})$ [1] à éléments dans l'ensemble des parties d'un ensemble arbitraire G (* désigne l'ensemble de ses opérations algébriques, $\cup$ la réunion ensembliste) la matrice de terme général

$$\bigcup_{k,*} m_{(\alpha k)} * m_{(k\beta)}$$

et puissance $n^{ème}$ $\overset{\cup}{*} M^n$ ou M^n de M dans $\mathcal{P}(G)$ la matrice $M^{n-1} M$ si les opérations * sont pseudo-associatives, sinon la matrice $M^{n-1} M \cup M M^{n-1}$.

Définissons la pseudo-fermeture transitive $\overset{\cup}{*} \overline{M}$ ou $\overline{M}$ de M par

$$\overline{M} = M \cup M^2 \cup M^3 \cup \ldots = \bigcup_{j=1}^{\infty} M^j$$

Elle sera dite exister (ou se stabiliser ou stationnaire ou finie) s'il existe un entier J fini tel que

$$\bigcup_{j=1}^{J} M^j = \bigcup_{j=1}^{J+1} M^j$$

sinon, $\overset{\cup}{*}_{j=1} \overline{M}$ désignera la limite de $\bigcup_{j=1}^{\infty} M^j$.

La pseudo-fermeture transitive d'une matrice coïncide avec la fermeture transitive dans le cas où * désigne une seule opération associative.

[1] $m_{(\alpha\beta)}$, l'élément dans la case (α, β) de M, n'est pas obligatoirement affecté du couples d'indices (α, β) comme le serait $m_{\alpha\beta}$.

b. Pseudo-fermeture transitive d'une matrice définie sur l'ensemble des parties d'un ensemble vérifiant les conditions I, M et D.

Théorème.

Les éléments de la pseudo-fermeture transitive réduite d'une matrice M(A) construite à partir d'une partie A d'un ensemble G d'éléments satisfaisant les conditions I, M et D, sont les éléments maximaux de A^* :

$$\text{Max } \overline{M(A)} = M(\text{Max } A^*)$$

Rappelons que $M(\text{Max } A^*)$ désigne l'écriture matricielle de Max A^* (voir sa construction en I-1 a1).

Une condition nécessaire et suffisante pour que Max $\overline{M(A)}$ existe est que A^* vérifie la condition de finitude B_3 .

Démonstration :

Max $\overline{M(A)}$ = $M(\text{Max } A^*)$ est évident puisque les éléments de $\overline{M(A)}$ sont les éléments de A^*. La seconde partie est immédiate ; elle peut aussi être considérée comme une conséquence de l'algorithme M1!

Etant donnée une matrice M dont les éléments sont des parties d'un ensemble à opérations isotones , le théorème précédent permet le calcul de Max $\overline{M}$; en effet montrons qu'il est possible d'associer à M une algèbre vérifiant les conditions I, M et D et dont les éléments maximaux sont les éléments de Max $\overline{M}$.

c. Algèbre $< A(M), \leq, * >$ associée à une matrice carrée définie sur l'ensemble des parties d'un ensemble à opérations isotones.

1. Définition.

Soit H un ensemble à opérations isotones (notées.), M une matrice carrée d'ordre I définie sur $\mathcal{P}(H)$; définissons $< A(M), \leq , * >$ ou l'algèbre associée à M de la façon suivante :

A(M) est l'ensemble des éléments de M indicés par le couple ordonné du numéro α de leur ligne et du numéro β de leur colonne (a, un élément de la $\alpha^{ème}$ ligne et de la $\beta^{ème}$ colonne de M, donne l'élément $a_{\alpha\beta}$ de A(M), où $\alpha, \beta = 1, 2, \ldots, I$;

$A(M) \subset \mathcal{P}(H) \times I \times I)$;

les * - composés de deux de ses éléments $b_{\alpha\beta}$ et $c_{\gamma\delta}$ sont nuls si $\beta \neq \gamma$, sinon ils sont égaux aux éléments b.c affectés des indices (α, δ) :

$$b_{\alpha\beta} * c_{\beta\delta} = (b.c)_{\alpha\delta}$$

cette égalité est un égalité générale dès que . représente plus d'une opération.

l'ordre $\leq$ dans $A(M)^*$ est ainsi défini : deux éléments d'indices différents sont incomparables ; deux éléments de mêmes indices $b_{\alpha\beta}$ et $c_{\alpha\beta}$ ont pour ordre celui de b et c dans H.

2. Propriétés de $< A(M), \leq , * >$

$< A(P), \leq , * >$ vérifie les conditions

(I) ou $b_{\alpha\beta} \leq c_{\alpha\beta} \Rightarrow b_{\alpha\beta} * d_{\gamma\delta} \leq c_{\alpha\beta} * d_{\gamma\delta}$ et $d_{\gamma\delta} * b_{\alpha\beta} \leq d_{\gamma\delta} * c_{\alpha\beta}$

en effet, si $<_H$ est la relation d'ordre dans G, on a, si $\beta = \gamma$:

$b_{\alpha\beta} \leq c_{\alpha\beta} \iff b \underset{H}{\leq} c \Rightarrow bd \underset{H}{\leq} cd \iff (bd)_{\alpha\delta} \leq (cd)_{\alpha\delta} \iff b_{\alpha\beta} * d_{\gamma\delta} \leq c_{\alpha\beta} * d_{\gamma\delta}$;

si $\beta \neq \gamma$, $b_{\alpha\beta} * d_{\gamma\delta} \leq c_{\alpha\beta} * d_{\gamma\delta}$ se réduit à $0 \leq 0$. On montrerait semblablement la deuxième implication.

(M) , (D) , (E)

(C) en considérant à la place de $*$, les I opérations algèbriques $*_i$

($i = 1, 2, \ldots, I$), ainsi définies

$$\begin{cases} b_{\alpha\beta} *_i c_{\gamma\delta} = (bc)_{\alpha\delta} & \text{si } \beta = \gamma = i \\ b_{\alpha\beta} *_i c_{\gamma\delta} = 0 & \text{sinon} \end{cases}$$

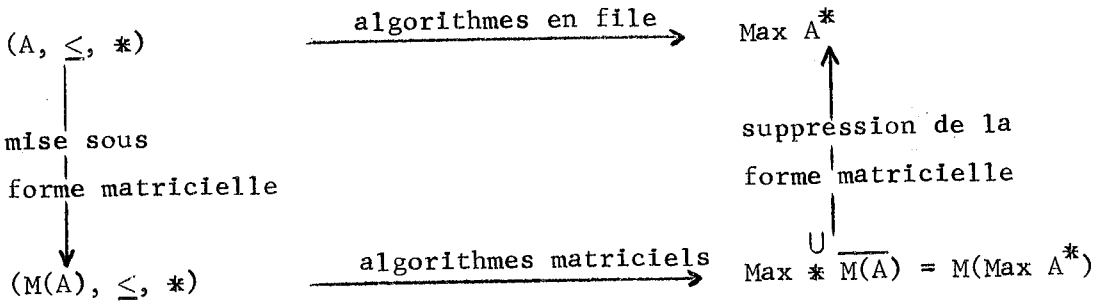
d. Equivalence entre $\underset{U}{\text{Max}} . \overline{M}$ et les éléments maximaux de $\langle A(M), \leq, * \rangle$

1. Théorème.

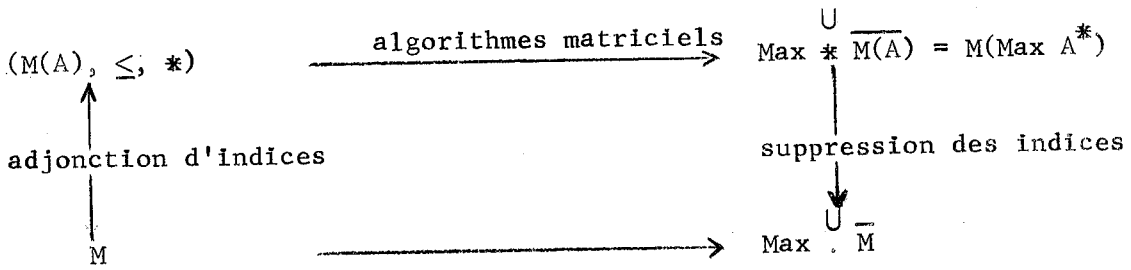
Les éléments d'une case (α, β) de $\underset{U}{\text{Max}} . \overline{M}$ sont les éléments maximaux d'indices (α, β) de l'algèbre $\langle A(M), \leq, * \rangle$ associée à M, après enlèvement des indices. Une condition nécessaire et suffisante d'existence de $\underset{U}{\text{Max}} . \overline{M}$ est que les éléments maximaux de l'algèbre associée soient en nombre fini et puissent être obtenus à partir de A par un nombre fini d'application de $*$.

Démonstration .

Considérons le schéma de I-3 appliqué à $A = A(M)$:



Dans le passage de $M(A)$ à $\mathrm{Max}^{\cup}_{*}\,\overline{M(A)}$, les indices ne jouent aucun rôle puisqu'ils n'interviennent pas pour la valeur des $*$ - composés, leurs couples d'indices excepté. Les adjonctions et suppressions d'indices étant des opérations inversibles, on peut donc dresser le schéma suivant :



La première partie du théorème est démontrée.

La deuxième résulte de ce qu'une condition nécessaire et suffisante d'existence de $M(\mathrm{Max}\ A^{*})$ est que $< A(M), \leq, * >$ vérifie la condition B3 (théorème de b).

2. Cas particulier

Les éléments d'une case (α, β) de $\overset{\cup}{.}\,\overline{M}$ sont les éléments d'indices (α, β) de l'algèbre $< A(M), \leq, * >$ associée à M, après enlèvement des indices. Une condition nécessaire et suffisante d'existence de $\overset{\cup}{.}\,\overline{M}$ est que

l'algèbre associée ait un nombre fini d'éléments.

Cette proposition est une conséquence immédiate du théorème précédent en prenant pour relation d'ordre dans H la relation d'ordre la plus grossière : deux éléments non nuls et distincts de H sont incomparables.

III.2. Fermeture transitive d'une matrice carrée définie sur un gerbier.

Rappelons qu'un gerbier [DUBREIL-JACOTIN, LESIEUR, CROISOT] est un ensemble E muni de deux opérations internes partout définies :

- la première, associative, commutative et idempotente, sera appelée somme et notée + ;
- la deuxième associative et distributive à droite et à gauche par rapport à la somme sera appelée produit et notée . ;

un tel gerbier sera noté (E, +, .).

Les définitions données dans III-1a s'étendent immédiatement aux matrices définies sur un gerbier ; la notion de pseudo-fermeture transitive se réduit à celle de fermeture transitive.

a. Théorème.

Soit M une matrice carrée définie sur un gerbier (E, +, .). Un élément $\overline{m}_{(\alpha\beta)}$ de sa fermeture transitive $\overset{+}{M}$ est la somme des éléments d'indices (α, β) de l'algèbre $< A(M), \leq, * >$ associée à M, après anlèvement des indices ; une condition suffisante d'existence de $\overset{+}{M}$ est que l'algèbre associée ait un nombre fini d'éléments.

Si l'ordre de $< A(M), \leq , * >$ est équivalent à une loi d'absorption dans (E, +, .)

$$b_{\alpha\beta} \leq c_{\alpha\beta} \iff b + c = c$$

alors une condition nécessaire et suffisante d'existence de $\overset{+}{\cdot}\,\overline{M}$ est que les éléments maximaux de l'algèbre associée soient en nombre fini et puissent être obtenus à partir de A(M) par un nombre fini d'applications de $*$; un de ses éléments $\overline{m}_{(\alpha\beta)}$ est la somme des éléments maximaux d'indices (α, β) de $< A(M), \leq , * >$, après enlèvement des indices.

Démonstration :

La comparaison des fermetures transitives de M dans (E, +, .)

$$\overset{+}{\cdot}\,\overline{M} = M + \overset{+}{\cdot}\,M^2 + \overset{+}{\cdot}\,M^3 + \ldots$$

et dans l'ensemble des parties du monoïde {éléments de M}$^\circ$ affecté des relations identiques s'il en existe dans lesquelles $\cup$ est substitué à + :

$$\overset{\cup}{\cdot}\,\overline{M} = M \cup \overset{\cup}{\cdot}\,M^2 \cup \overset{\cup}{\cdot}\,M^3 \cup \ldots$$

montre que $\overset{+}{\cdot}\,\overline{M}$ est obtenue à partir de $\overset{\cup}{\cdot}\,\overline{M}$ en remplaçant $\cup$ par + ; la première partie de théorème se déduit alors du cas particulier précédent (III-1d).

Si l'ordre $\underset{\overline{M}^\circ}{\leq}$ dans {éléments de M}$^\circ$ est équivalent à une loi d'absorption dans (E, +, .)

$$b \underset{\overline{M}^\circ}{\leq} c \iff b + c = c$$

alors le théorème de III-1d est applicable.

b. Conclusion.

Le problème de l'existence et du calcul de la fermeture transitive $\overset{+}{\cdot}\overline{M}$ d'une matrice carrée M définie sur un gerbier (ou plus généralement de Max $\overset{+}{\cdot}\overline{M}$) est donc équivalent à celui de la recherche des éléments (maximaux) du monoïde $< A(M), \leq, * >$ associé à la matrice M .

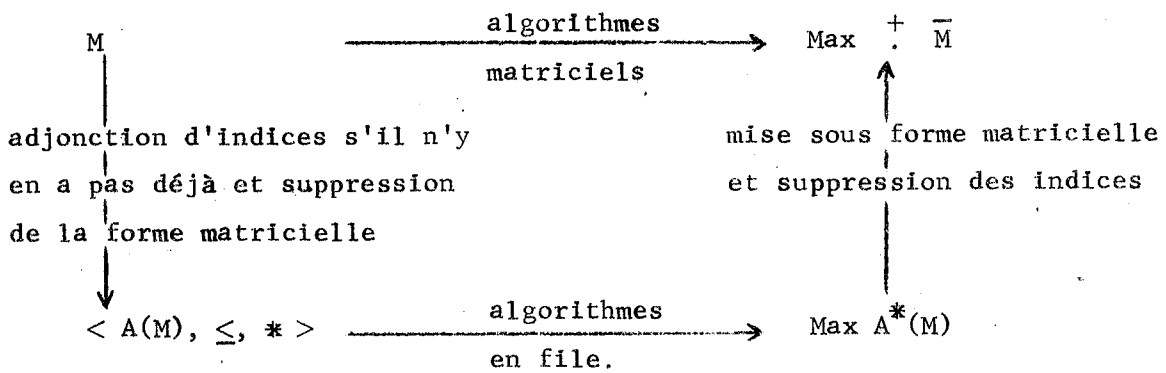
1. Si $< A(M), \leq, * >$ vérifie la condition B3, Max $\overset{+}{\cdot}\overline{M}$ existe et peut-être calculé

soit à l'aide des algorithmes matriciels étudiés en I , appliqués directement à M en changeant $\cup$ en + (l'algorithme le meilleur est CM),

soit à l'aide des algorithmes en file (chapitre 4 et remarque 3 de I-2):

appliqués à $< A(M), \leq, * >$ en changeant $\cup$ en + pour des éléments de mêmes indices, ils donnent un élément maximal d'indices (α, β) qui, après enlèvement des indices, est l'élément de la case (α, β) de Max $\overset{+}{\cdot}\overline{M}$

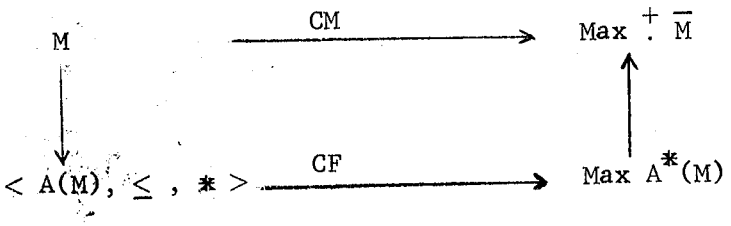
Schématiquement, on peut ainsi figurer ces deux processus :



Une application aux pseudo-treillis distributifs est donnée dans le paragraphe suivant .

2. Si $< A(M), \leq , * >$ ne vérifie pas la condition B3, alors $\text{Max}^{+}_{\cdot} \overline{M}$ n'existe pas ; cependant l'algorithme CM ou CF peut être utilisé pour localiser et décomposer le phénomène de non-stabilisation, à condition d'introduire les éléments symboliques nécessaires.

Le schéma précédent devient :



Une première application est la détermination de la fermeture transitive de matrices définies sur une algèbre d'expressions régulières : CM devient l'algorithme de [McNAUGHTON, YAMADA]. Un exemple pourrait être celui de II-1c où a, b, c, d, e, f désigneraient des éléments d'une algèbre d'expressions régulières.

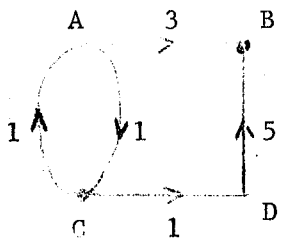
Une autre application [PAIR, DERNIAME], [VEILLON], est la détermination de la plus longue distance, des plus longs chemins, et du nombre de chemins d'un sommet à un autre d'un graphe.

Les structures algèbriques, respectivement (réels ≥ 0, max, +), (chemins, le plus long de deux chemins, concaténation), (entiers naturels, +, x), sont des gerbiers (mais non des pseudo-treillis distributifs). Les algorithmes CM ou CF permettent donc de calculer leur éléments maximaux, à condition

d'introduire les éléments symboliques suivants : $+\infty$, ensemble des plus longs circuits passant par le sommet x_i et par des sommets d'indices inférieurs à i avec $i = 1, 2, \ldots, I, +\infty$, respectivement.

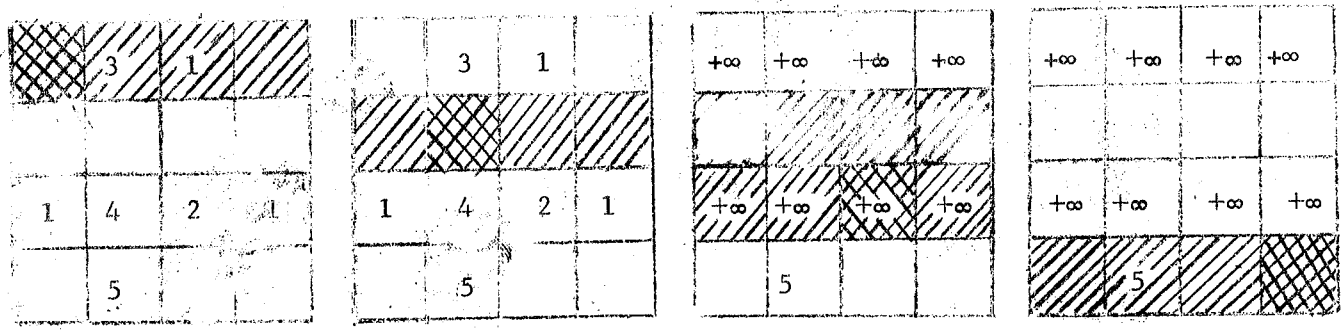
Les fermetures transitives de matrices définies sur ces structures existent (sans l'introduction d'éléments symboliques) dans le cas des graphes sans circuits et sans boucles.

Par exemple, recherchons la plus longue distance séparant tout couple de points du graphe suivant :



	-A	-B	-C	-D
A-		3	1	
B-				
C-	1			1
D-		5		

Les itérations successives de CM donnent :



Remarque 1.

Pratiquement, pour rechercher la fermeture transitive d'une matrice définie sur un gerbier, on peut d'abord appliquer CM ou CF, voir s'il est nécessaire d'introduire dans les cases de la diagonale principale (phase 1 des algorithmes) des éléments symboliques ; si cela est nécessaire, la fermeture n'existe pas.

Remarque 2.

La distributivité à droite et à gauche du produit par rapport à la somme n'est pas nécessaire pour la définition d'une fermeture transitive et la théorie précédente. Par exemple, si . est distributif à gauche par rapport à + , la puissance nème d'une matrice sera ainsi définie :

$$M^n = (\ldots(M(M\ M))).$$

L'existence d'une opération produit univoque n'est aussi pas nécessaire : . , représentant un ensemble d'opérations pseudo-associatives et distributives par rapport à la somme, définirait alors avec la somme une structure algèbrique que l'on pourrait appeler pseudo-gerbier. Les résultats précédents restent valables pour la pseudo-fermeture transitive d'une matrice carrée définie sur un pseudo-gerbier.

III.3. Application à la fermeture transitive d'une matrice carrée définie sur un pseudo-treillis distributif.

a. Existence de la fermeture.

1. Un pseudo-treillis distributif (E, +, .) [BENZAKEN 1]ou gerbier quasi-entier [DUBREIL-JACOTIN, LESIEUR, CROISOT] est un ensemble muni de deux

opérations internes + et . partout définies, et vérifiant les propriétés suivantes :

$$\begin{cases} + \text{ est associatif, commutatif, idempotent} \\ . \text{ est associatif} \\ . \text{ est distributif à droite et à gauche par rapport à } + \\ b + bc = b \text{ et } b + cb = b \text{ pour tous } b \text{ et } c \end{cases}$$

E admet un élément nul ($0 + a = a$, pour tout a).

2. Les éléments maximaux de l'algèbre $< A(M), \leq, * >$ associé à une matrice M définie sur un pseudo-treillis distributif $(E, +, .)$ ou sur le monoïde {éléments de M}* ordonné par $bc \underset{M^*}{\leq} b$ et $cb \underset{M^*}{\leq} b$, sont en nombre fini et peuvent être obtenus par un nombre fini d'applications de $*$ puisqu'ils sont de la forme :

$$b_{\beta\gamma} * c_{\gamma\delta} * \ldots * d_{\varepsilon\varphi}$$

où b, c, ..., d sont des éléments de M et où les indices $\beta, \gamma, \delta, \ldots, \varepsilon, \varphi$ sont tous différents, excepté peut-être β et φ ; en effet :

$$\begin{cases} b_{\beta\gamma} * c_{\gamma\gamma} = (bc)_{\beta\gamma} \leq b_{\beta\gamma} \\ b_{\beta\beta} * c_{\beta\gamma} = (bc)_{\beta\gamma} \leq c_{\beta\gamma} \end{cases}$$

Donc, en vertu du théorème de III-2a :

<u>La fermeture transitive $\overset{+}{.}\overline{M} = M + \overset{+}{.} M^2 + \overset{+}{.} M^3 + \ldots$ d'une matrice carrée M définie sur un pseudo-treillis distributif existe</u>.

Cette généralisation du théorème de LUNC a déjà été donnée dans [BENZAKEN 1], puis dans [TOMESCU 3] dans le cas où M est surunitaire et l'opération . est commutative, et dans [ROBERT, FERLAND].

Quel est le rang J à partir duquel $\overset{J}{\underset{j=1}{+}} \overset{.}{} M^j$ devient stationnaire ?

Les éléments maximaux de $< A(M), \leq , * >$ étant obtenus à partir d'au plus I-1 éléments de M si M est surunitaire, sinon à partir de I éléments au plus, on a :

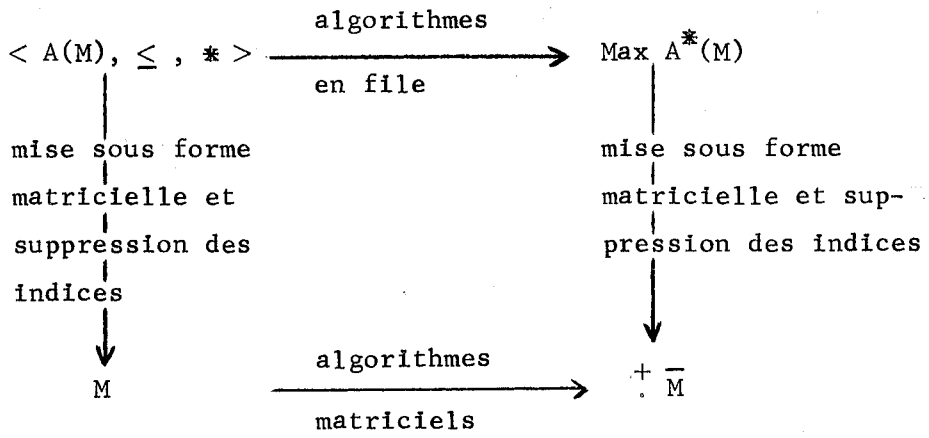
$$\overset{+}{.} \overline{M} = M + \overset{+}{.} M^2 + \ldots + \overset{+}{.} M^{I-1} \qquad \text{si M est surunitaire}$$

$$\overset{+}{.} \overline{M} = M + \overset{+}{.} M^2 + \ldots + \overset{+}{.} M^{I-1} + \overset{+}{.} M^{I} \qquad \text{sinon}$$

b. Exemples d'application.

Les problèmes énoncés dans II-2 d'existence de chemins, de plus courte distance, de plus courts chemins, de probabilité de passage d'un point à un autre, de capacité maximum peuvent être considérés comme des problèmes de fermeture transitive de matrices carrées définies respectivement sur les pseudo-treillis distributifs suivants ($\{0, 1\}$, addition booléenne, $\times$) , (R_+^∞ , min, +) , (chemins, le plus court chemin de deux chemins, concaténation) ou (chemins, le plus long de deux chemins, concaténation) , ($[0, 1]$, +, x), (R_+^∞ , max, min).

Il nous paraît souvent plus commode pour poser le problème de considérer d'abord le pseudo-treillis espace de l'information, les chemins jouant le rôle de fonction de localisation aux éléments du pseudo-treillis, puis de mettre le problème sous forme matricielle M , enfin d'appliquer les algorithmes matriciels pour obtenir $\overset{+}{.} \overline{M}$:



c. Algorithmes matriciels de calcul.

Ce sont les algorithmes de II-2b et en particulier de II-2c après substitution de $+$ à $\cup$.

1. Les algorithmes matriciels généraux M1', M2 et M2' donnent respectivement :

Algorithme M1'.

On calcule successivement :

$$N_2 = M + M^2$$

$$N_4 = N_2 + (N_2)^2$$

...

$$^{+}\bar{M} = N_{2^p} = N_{2^{p-1}} + (N_{2^{p-1}})^2$$

où p est le plus petit entier tel que $2^p \geq I$.

Algorithme M2.

- Pour les I^2 couples (α, β) $(\alpha, \beta = 1, 2, \ldots, I)$,

on calcule $\overset{I}{\underset{i=1}{+}} a_{(\alpha i)} \cdot a_{(i\beta)}$, ce qui donne un élément

$a_{(\alpha\beta)}$,

on ajoute $a_{(\alpha\beta)}$ à l'élément de mêmes indices de la matrice ;

- on itère jusqu'à stabilisation (au plus p fois avec $2^{p-1} < I \leq 2^p$).

Algorithme M2'.

- Pour $\alpha = 1, 2, \ldots, I$, on itère jusqu'à stabilisation (au plus I-1 fois) le processus suivant :

- pour $\beta = 1, 2, \ldots, I$, on calcule $\sum_{i=1}^{I} a_{(\alpha i)} \cdot a_{(i\beta)}$, ce qui donne un élément $a_{(\alpha\beta)}$; dès qu'un tel $a_{(\alpha\beta)}$ a été formé, on l'ajoute à l'élément dans la case (α, β) de la matrice.

A la première stabilisation, on obtient la première ligne de la fermeture transitive cherchée, à la seconde stabilisation la deuxième ligne,..., à la $I^{ème}$ la $I^{ème}$ ligne.

Appliqué à la recherche des plus courtes ou des plus longues distances entre le point x_1 et tout autre point $(\alpha = 1)$, on obtient les algorithmes de FORD et de BELLMAN-KALABA.

2. Au lieu de $.$, considérons les I opérations algèbriques $\underset{i}{.}$ $(i = 1, 2, \ldots, I)$ ainsi définies :

$$\begin{cases} b_{(\alpha\beta)} \underset{i}{.} c_{(\gamma\delta)} = (b.c)_{(\alpha\delta)} & \text{si } \beta = \gamma = i \text{ et si } bc \neq 0 \\ \underset{i}{.} \text{ non défini} & \text{si } \beta \neq i \text{ ou } \gamma \neq i \text{ ou } bc = 0 \end{cases}$$

Elles vérifient :

$$(a_{(\alpha\beta)} \underset{i}{.} b_{(\gamma\delta)}) \underset{j}{.} c_{(\varepsilon\varphi)} = a_{\alpha\beta} \underset{i}{.} (b_{\gamma\delta} \underset{j}{.} c_{\varepsilon\varphi})$$

puisque $.$ est associatif, soit la condition (C).

Algorithme CM

- Pour $i = 1, 2, \ldots, I$,
- pour les $(I - 1)^2$ couples (α, β) $(\alpha, \beta = 1, 2, \ldots, I$ avec $\alpha \neq i, \beta \neq i)$

on calcule $a_{(\alpha i)} a_{(i\beta)}$ ce qui donne un élément $a_{(\alpha\beta)}$,

on ajoute $a_{(\alpha\beta)}$ à l'élément de la case (α, β) de la matrice.

Démonstration :

Pour un i fixé, les calculs se font à partir des éléments de la $i^{ème}$ colonne et de la $i^{ème}$ ligne, $a_{(\alpha i)}$ et $a_{(i\beta)}$. Ces éléments pivots $a_{(\alpha i)}$ et $a_{(i\beta)}$ sont invariants puisque $a_{\alpha i} \underset{i}{*} a_{ii} \leq a_{\alpha i}$ et $a_{ii} \underset{i}{*} a_{i\beta} \leq a_{i\beta}$; il

est donc inutile, quand $\alpha = \beta = I$, d'itérer sur α et β pour le même i.

CM, CM1' et CM2 sont identiques à cause de l'invariance des éléments pivots.

Algorithme M2'C

- Pour $\alpha = 1, 2, \ldots, I$
- pour $i = 1, 2, \ldots, I$ avec $i \neq \alpha$
- pour les (I-1) couples (α, β) $(\beta = 1, 2, \ldots, I$ avec $\beta \neq i)$

on calcule $a_{(\alpha i)} \cdot a_{(i\beta)}$, ce qui donne $a_{(\alpha\beta)}$,

on ajoute $a_{(\alpha\beta)}$ à l'élément de mêmes indices de la matrice.

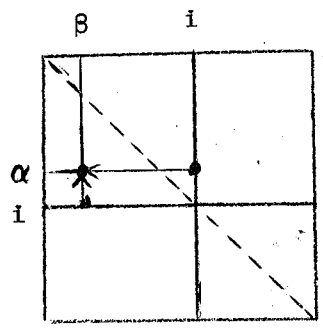
Remarque.

Pour obtenir l'algorithme CM, on pourrait introduire, au lieu d'opérateurs binaires $*_i$, des opérations unaires o_i $(i = 1, \ldots, I)$ comme il est fait dans [BENZAKEN 1] par exemple ; elles sont définies sur les matrices carrées d'ordre I $M = \{m_{(\alpha\beta)}\}$ à élément $m_{(\alpha\beta)}$ dans un pseudo-treillis de la façon suivante :

$$M^{*_i} = o_i \; M$$

où

$$m^{*_i}_{(\alpha\beta)} = m_{(\alpha\beta)} + m_{(\alpha i)} m_{(i\beta)}$$



Les opérations o_j ont les propriétés de

- commutativité : $\overset{o}{i}\ \overset{o}{j} M = \overset{o}{j}\ \overset{o}{i} M$ (la condition C est donc vérifiée)

- idempotence : $\overset{oo}{ii} M = \overset{o}{i} M$

De l'obtention de la fermeture d'une matrice M par l'application un nombre fini de fois des opérations $\overset{o}{i}$, on déduit :

$$M + M^2 + \ldots = \overset{o}{1} \ldots \overset{oo}{12} \ldots \overset{oo}{23} \ldots \overset{o}{n} M \text{ , d'après l'algorithme C}$$

$$= \overset{ooo}{123} \ldots \overset{o}{n} M \text{ , d'après l'idempotence des } \overset{o}{i} .$$

3. Remarques.

L'algorithme M1' très classique est par exemple dans [MAGHOUT], [BENZAKEN 1] ; M2 est dans [BENZAKEN 1] et le cas particulier des matrices booléennes est dans [TOMESCU 1] ; M2' est dans [TOMESCU 2] et la généralisation de l'algorithme de ROY-WARSHALL CM est dans [BENZAKEN 1], [ROBERT, FERLAND], et, dans le cas où M est surunitaire et le produit commutatif, dans [TOMESCU 3].

4. Cas particulier de la fermeture transitive d'une matrice surunitaire $M_1 = 1 + M$ ou fermeture transitive forte de M [KUNTZMANN 1] chapitre II, p. 19, où 1 désigne la matrice unitaire d'ordre I sur (E, +, .). Remarquons d'abord que la fermeture transitive forte de M, définie par $1 + M + M^2 + M^3 + \ldots$ est égale à la fermeture transitive de $1 + M$, puisque

$$(1 + M)^{\infty} = 1 + M + M^2 + M^3 + \ldots$$

Les résultats précédents se simplifient en remarquant que :

$$\sum_{k=1}^{K} M_1^k = M_1^K$$

$$\overset{+}{\cdot} \bar{M}_1 = M_1 + \overset{+}{\cdot} M_1^2 + \ldots + \overset{+}{\cdot} M_1^{I-1}$$

(Voir par exemple [BENZAKEN 1] p. 148, [ROBERT, FERLAND] p. 76).

d. Algorithmes en file de calcul de la fermeture transitive d'une matrice carrée définie sur un pseudo-treillis distributif. Application à la fermeture transitive d'un graphe ou d'une relation binaire.

1. Ces algorithmes sont les algorithmes généraux en file ou mieux les algorithmes matriciels disposés en file comme CM (voir remarque 3 de I-2); les éléments de mêmes indices sont groupés, mais ces groupes sont disposés en file et non en matrice) appliqués au monoïde $< A(M), \leq , * >$; on peut à tout instant remplacer des éléments de mêmes indices par leurs sommes (sommes des éléments après suppression des indices, auxquelles on ajoute de nouveau des indices) ; en fin d'algorithme, on déduit la matrice fermeture en affectant à chacune de ses cases (α, β) la somme des éléments d'indices (α, β) après enlèvement de leurs indices.

2. Les algorithmes en file peuvent être plus rapides que les algorithmes matriciels. En effet, les algorithmes matriciels utilisent un support (à I cases pour les problèmes en corrélation avec les chemins d'un graphe à I sommets ayant pour origine ou pour extrémité un point donné ; à I^2 cases pour les problèmes en corrélation avec plus généralement les chemins quelconques d'un graphe à I sommets ou pour la fermeture d'une matrice I x I) avec ses

avantages (et ses inconvénients) intrinsèques de maniement des indices ; par contre, les algorithmes en file construisent de nouveaux éléments non à partir de cases peut-être vides d'un cadre préconçu, mais à partir des éléments existants ; l'efficacité relative de ces algorithmes dépendra de la valeur du rapport

$$\frac{\text{Nombre de cases du support de l'algorithme matriciel}}{\text{Nombre d'opérations * utiles et nombre d'opérations * inutiles}}$$

En général, pour une matrice très creuse, la file sera courte, le nombre d'opérations * utiles sera faible (la file restera courte), le nombre d'opérations * inutiles sera faible par rapport au nombre d'opérations * à effectuer sur les cases vides dans un algorithme matriciel, un algorithme en file sera plus rapide qu'un algorithme matriciel ; inversement pour une matrice assez pleine, un algorithme matriciel sera plus rapide qu'un algorithme en file. Quant à la place en mémoire, celle nécessaire à un algorithme en file est au plus égale à celle nécessaire à un algorithme matriciel.

Application en a été faite à l'analyse syntaxique, par exemple dans [COURTIN], [GRIFFITHS]. Deux autres applications vont maintenant être détaillées.

3. En particulier, considérons un graphe $(X = \{x_1, x_2, \ldots, x_\alpha, \ldots, x_\beta, \ldots, x_I\}, \Gamma)$ et sa matrice booléenne associée ; rappelons que c'est une matrice à I lignes et à I colonnes dont chaque case sur sa diagonale principale (α, α) contient 1 et dont chaque autre case (α, β) contient 1 si le graphe a un arc $x_\alpha x_\beta$, sinon 0. Elle donne l'existence des chemins d'un point à un autre.

Considérons le graphe $(X, \overline{\Gamma})$ fermeture transitive du graphe (X, Γ)

où $\overline{\Gamma}$ est la fermeture transitive de Γ [BERGE 1] p. 4, c'est-à-dire l'application de X dans X définie par

$$\overline{\Gamma}x_i = x_i \cup \Gamma x_i \cup \Gamma^2 x_i \cup \Gamma^3 x_i \cup \ldots$$

La matrice associée à $(X,\overline{\Gamma})$ ou matrice donnant l'existence des chemins d'un point à un autre du graphe (X,Γ) est la fermeture transitive de la matrice associée à (X,Γ) ou d'après , c4, la fermeture transitive forte de la matrice ayant 0 dans chaque case de la diagonale principale, égale ailleurs à la matrice associée à (X,Γ).

Les algorithmes précédents de fermeture de matrices s'appliquent, et les algorithmes en file prennent en particulier la forme suivante. Puisqu'en dehors de l'élément 0 (ne figurant pas dans un algorithme en file), il n'existe que l'élément 1, chaque élément 1 de la matrice associée à un graphe dans une case (α,β) est caractérisé par la donnée du monôme ordonné $\alpha'\beta$ ou $\beta\alpha'$ (au lieu de $1_{\alpha\beta}$) ; les règles de manipulation dans M de $*$ (III-1 c1)

$$\begin{cases} b_{\alpha\beta} * c_{\gamma\delta} = (bc)_{\alpha\delta} & \text{si} \quad \beta = \gamma \\ * \quad \text{non défini} & \text{si} \quad \beta \neq \gamma \end{cases}$$

ou de $*_i$ (III-1 c 2)

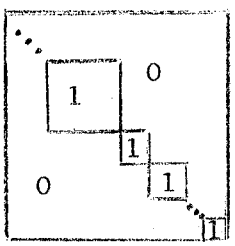
$$\begin{cases} b_{\alpha\beta} *_i c_{\gamma\delta} = (bc)_{\alpha\delta} & \text{si} \quad \beta = \gamma = i \\ *_i \quad \text{non défini} & \text{si} \quad \beta \neq i \text{ ou } \gamma \neq i \end{cases}$$

s'identifient respectivement aux règles de formation des consensus ou des consensus par rapport à la variable i (5-I-2c) à l'exception près suivante : $\alpha\beta' * \beta\alpha' = \alpha\alpha'$ alors que le consensus n'existe pas. Puisque les circuits ne contribuent pas à la fermeture transitive d'un graphe, on peut utiliser exclusivement les algorithmes de recherche des monômes premiers d'une fonction booléenne (chapitres 3, 5-I, 7-V et VI), puis ajouter les éléments diagonaux $\alpha\alpha'$; on retrouve les résultats de [KUNTZMANN 1] chapitre II, p. 19.

En étendant l'existence du consensus à des monômes de la forme $\alpha\beta'$ et $\alpha'\beta$ ou en traitant séparément les éléments diagonaux, cette méthode d'utilisation des monômes booléens à deux lettres complémentée et non complémentée est applicable à la fermeture transitive d'une matrice booléenne arbitraire.

Une application est la recherche des sous-graphes fortement connexes maximaux d'un graphe (ils partitionnent l'ensemble des sommets du graphe) : en effet on calcule la fermeture transitive de la matrice booléenne associée au graphe et on la symétrise en enlevant les 1 nécessaires ; c'est la méthode de FOULKES ; l'algorithme M2' appliqué à la fermeture transitive donne à peu près l'algorithme de MALGRANGE [KAUFMANN 1] p. 213 pour trouver les sous-graphes fortement connexes maximaux.

4. Pour la fermeture transitive d'un graphe symétrique et d'une matrice booléenne symétrique, il est préférable d'utiliser une méthode spécifique : à chaque élément $1_{\alpha\beta}$ est associé le monôme $\alpha\beta$, et les monômes obtenus sont traités à l'aide des algorithmes donnant la réunion de relations d'équivalence (5-II) ; la fermeture est alors donnée non par tous ses carrés atomiques mais par ses carrés maximaux ; sa forme, à une permutation près des lignes et des colonnes est :



CHAPITRE 7

LES DEVELOPPEMENTS

I - GENERALITES SUR LES DEVELOPPEMENTS

I.1. Définition d'un développement.

Etant données n parties (distinctes ou non) d'ensembles ordonnés ou facteurs

$$F_1 = a_{1,1} \cup a_{1,2} \cup \ldots \cup a_{1,M}$$

$$F_2 = a_{2,1} \cup a_{2,2} \cup \ldots \cup a_{2,Q}$$

...

$$F_n = a_{n,1} \cup a_{n,2} \cup \ldots \cup a_{n,R}$$

et une opération n-aire isotone o, développer $o(F_1, F_2, \ldots, F_n)$, c'est déterminer pour tout n-uplet $(a_{1,m}, a_{2,q}, \ldots, a_{n,r})$ de $F_1 \times F_2 \times \ldots \times F_n$ leur o-composé $o(a_{1,m}, a_{2,q}, \ldots, a_{n,r})$ s'il existe et ne garder que les o-composé maximaux.

Nous considérerons maintenant le cas où, quels que soient les facteurs $F_1, F_2, \ldots, F_n$ et quelle que soit l'opération o, il existe (au moins) une arborescence (bifurcante et indépendante du n-uplet $(a_{1,m}, a_{2,q}, \ldots, a_{n,r})$) $\mathscr{A}$ telle que

$$o(a_{1,m}, a_{2,q}, \ldots, a_{n,r}) = \mathscr{A}^* (a_{1,m}, a_{2,q}, \ldots, a_{n,r})$$

pour tout $m,q,\ldots,r$

où $*$ est un ensemble fini d'opérations binaires et isotones.

Nous supposons donc que o est réalisable à l'aide d'opérations binaires et nous pouvons parler indifféremment du développement de $(F_1, F_2, \ldots, F_n)$ ou du développement de $\mathscr{A}^* (F_1, F_2, \ldots, F_n)$

I.2. Les développements sont des algorithmes de recherche d'éléments maximaux.

a. Développer o $(F_1, F_2, \ldots, F_n)$ c'est par exemple rechercher les éléments maximaux de $(F_1 \times F_2 \times \ldots \times F_n , o, \leq)$

où o est considéré comme une opération unaire définie sur l'ensemble produit $F_1 \times F_2 \times \ldots \times F_n$,

où l'ensemble G dans lequel on opére est

$$F_1 \times F_2 \times \ldots \times F_n \cup o (F_1, F_2, \ldots, F_n),$$

soit la réunion de l'ensemble de départ $A = F_1 \times F_2 \times \ldots \times F_n$ et de son image par o A°,

où $\leq$ est ainsi définie : sa restriction à A est l'ordre de A, sa restriction à A° est l'ordre de A°, deux éléments de A et A° ne sont pas comparables.

b. Développer $\mathcal{A}^*(F_1, F_2, \ldots, F_n)$ est aussi un algorithme de recherche d'éléments maximaux

en prenant pour ensemble de départ $F_1 \cup F_2 \cup \ldots \cup F_n$,

en prenant pour fonction de localisation l'application l suivante (définie sur des arborescences et à valeurs des arborescences) :

$$\begin{cases} l\,(a_{i,j}) = i & \text{quels que soient } i = 1, 2, \ldots, n \text{ et } j \\ l\,(b \underset{i}{*} c) = l(b) \underset{i}{*} l(c), \end{cases}$$

en attachant à chaque sommet s de $\mathcal{A}$ (excepté les racines) une opération $\underset{s}{*}$ identique quand elle est définie à l'opération $\underset{i}{*}$ attachée à s, mais reinstreinte de la façon suivante :

$b \underset{s}{*} c$ est défini seulement si $l(b) \underset{i}{*} l(c) = \mathcal{B}^*_s(F_1, F_2, \ldots, F_n)$

où $\mathcal{B}^*_s(F_1, F_2, \ldots, F_n)$ est la sous-arborescence de

$\mathcal{A}^*(F_1, F_2, \ldots, F_n)$ de sommet s,

en prenant pour ordre entre deux éléments en lesquels la fonction de localisation a la même valeur, leur ordre dans G ; sinon, deux éléments sont incomparables.

I.3. Algorithmes de développement.

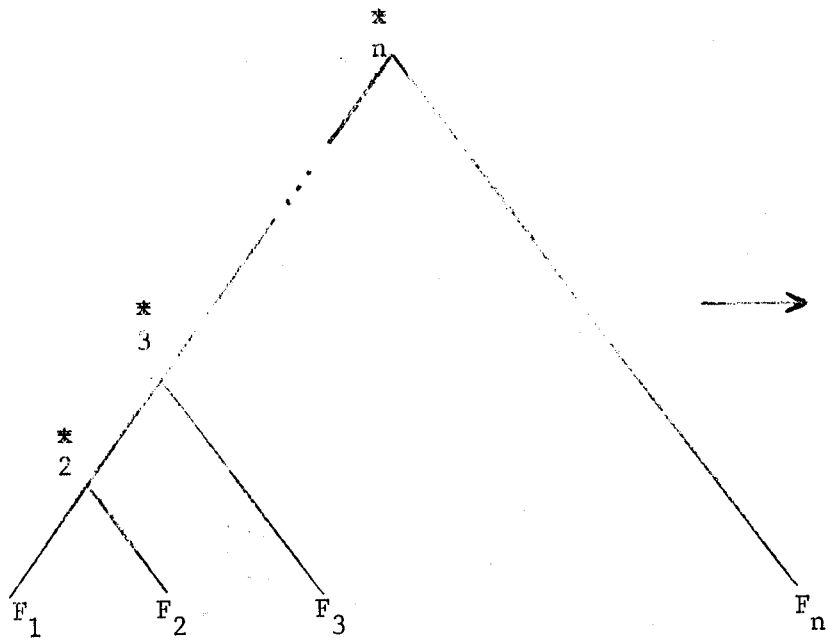
Après d'éventuelles réductions de facteurs ou à l'intérieur de facteurs, possibles à cause de l'isotonie de o, le développement de

$o(F_1, F_2, \ldots, F_n) = \mathcal{A}^*(F_1, F_2, \ldots, F_n)$

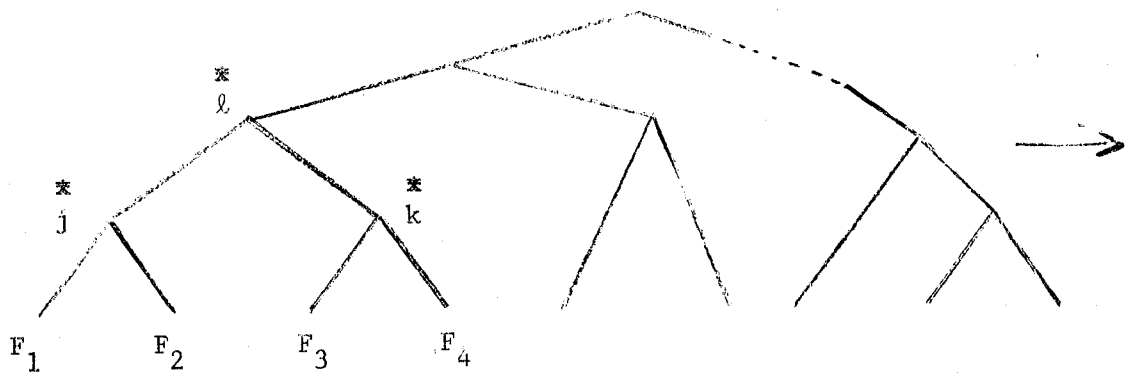
peut se faire de deux façons :

a. un développement progressif : chaque facteur est appelé une seule fois, l'ordre d'appel étant déterminé par $\mathcal{D}$ (c'est un algorithme récursif au sens de 4-II-2, à n récurrences). Les résultats obtenus à chaque récurrence sont déterminés globalement, en particulier tous les éléments maximaux sont déterminés en parallèle lors de la dernière récurrence.

Par exemple, si $\mathcal{A}$ est une arborescence bifurcante de la forme :



on développera $F_1 \underset{2}{*} F_2$, puis le résultat et F_3 sont composés par l'intermédiaire de $\underset{3}{*}$ ... ; si $\mathcal{A}$ est une arborescence bifurcante de la forme :



on développera $F_1 \underset{j}{*} F_2$, $F_3 \underset{k}{*} F_4$, ... puis leurs résultats deux par deux ...

b. un <u>développement différentié</u> ou recherche répétée d'un seul élément maximal c'est-à-dire brutalement :

- l'énumération de tous les n-uplets, en prenant un terme dans chacun des facteurs de toutes les façons possibles ; cette première phase peut-être considérée comme une donnée immédiate, théoriquement, sinon pratiquement à cause du nombre des combinaisons possibles ;
- la déduction, à partir de chaque n-uplet, de l'o-composé correspondant ;
- la réduction des o-composés inférieurs à d'autres globalement ou au fur et à mesure de leur formation.

L'énumération des n-uplets peut-être faite lexicographiquement de façon efficace quand les facteurs F_1, F_2, ..., F_n ont de nombreux éléments communs et quand la valeur de $o(a_{1,m}, a_{2,q}, \ldots, a_{n,r})$ est indépendante de l'ordre des éléments $a_{1,m}$, $a_{2,q}$, ..., $a_{n,r}$.

Quand les o-composés sont maximaux et distincts, la phase de réduction devient sans objet et un algorithme différentié à alors l'avantage sur les algorithmes progressifs de fournir des éléments du développement en cours de route.

c. <u>Remarque</u>.

Les algorithmes de développement, à cause de leur simplicité, sont soit des algorithmes de recherche d'éléments maximaux très primitifs (c'est souvent le cas lorsque les ensembles F_1, F_2, ..., F_n sont identiques ; d'ailleurs, pour toute recherche d'éléments maximaux, on peut toujours utiliser des développements et des réductions), soit au contraire des algorithmes de recherche d'éléments maximaux très efficaces (c'est le cas usuel pour des ensembles F_1, F_2, ..., F_n différents ; on utilise le fait que les opérations sont peu définies). Dans cette optique d'efficacité des développements, on

peut remarquer que l'ensemble $(A, *, \leq)$ associé à $\mathcal{A}^*(F_1, F_2, \ldots, F_n)$ dans 2-b vérifie la condition C et admet en conséquence, pour algorithme particulier de recherche d'éléments maximaux, tout développement progressif utilisant la structure de l'arborescence $\mathcal{A}$.

Ces impressions seront par la suite confirmées : une application brutale d'un algorithme de développement peut-être très lourde (voir en particulier II-1) ; par contre l'application répétée d'algorithmes de développement sera avantageusement comparable en VI à l'application d'un seul algorithme de recherche d'éléments maximaux.

II - ENUMERATION DE CHEMINS DANS LES GRAPHES FINIS

Rappelons que nous avons déja donné des algorithmes fournissant de telles énumération (6-II-2) ; nous en énonçons maintenant d'autres, qui sont des développements.

II.1. Enumération des chemins de longueur n obéissant à une propriété $\mathcal{P}$.

Ces chemins apparaissent comme le produit (suivant les opérations définies en 6-II-1b) de n arcs ; ils peuvent donc être obtenus en développant les produits

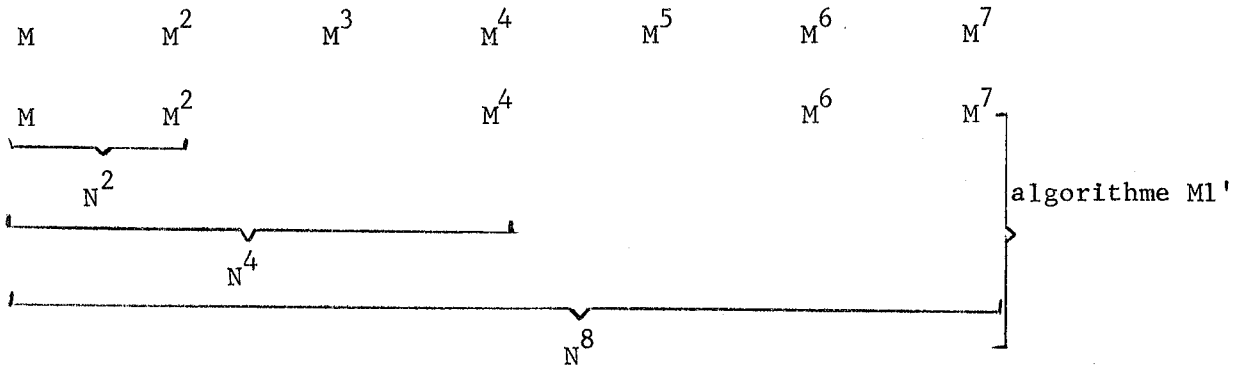
$$A^n = A \times A \times \ldots \times A \qquad \text{ou} \qquad M^n = M \times M \times \ldots \times M$$

A étant l'ensemble des arcs du graphe considéré, M sa forme matricielle, On peut par exemple énoncer :

On part de la forme matricielle M de A. On calcule successivement M^2, M^3, ... M^n , ou mieux M^2, M^4, M^8, ..., M^{2^p} (avec $2^p \leq n < 2^{p+1}$) , $M^{2^p + 2^q}$ (2^q étant la plus grande puissance de 2 inférieure ou égale à $n - 2^p$)., ... , M^n .

Ces deux algorithmes sont par exemple dans [PAIR, DERNIAME], [KAUFMANN 2]. Ils ont l'intérêt de fournir des chemins sans répétition et, à chaque itération, des chemins ayant le même nombre d'arcs.

Nous pouvons représenter ainsi la relation entre ces algorithmes et l'algorithme M1' , sur l'exemple du calcul de M^7 :



M1' a donc moins d'étapes ; s'il ne tient pas compte de la remarque 1 de 6-II-2b, il est redondant à cause de l'associativité de . : par exemple, il calculera les éléments à 4 arcs à la fois par

$$M^2 \overset{U}{.} M^2 \quad \text{et} \quad M \overset{U}{.} M^3.$$

Si n = I-1 (I est l'ordre de M), on trouve en particulier les chemins et circuits hamiltoniens.

Etudions ces algorithmes séparément :

a. Intéressons-nous au problème un peu plus général de la <u>détermination des chemins de n arcs d'origine un sous-graphe X_K de X</u> (qui peut être X ou un sommet X_k) <u>et d'extrémité tout point du graphe</u>. Ils sont obtenus en développant le produit

$$M(K, .) \times M \times M \times \ldots \times M = M(K, .) \times M^{n-1}$$

où M (K, .) est la ligne donc chacune des I cases est la réunion des cases appartenant à la même colonne et aux lignes de M dont les arcs ont pour origine un sommet de X_K.

Algorithme a.

On calcule successivement

$M(K, .)\ M,\ M(K, .)\ M^2, \ldots, M(K, .)\ M^{n-1}$

C'est l'algorithme de [PAIR, DERNIAME] p. 15. A la $i^{ème}$ itération ($i = 0, 1, 2, \ldots, n-2$) , on obtient une suite sans répétition de chemins de $i + 1$ arcs joignant X_K à tout point du graphe.

Remarque 1.

Le nombre d'opérations au niveau des cases matricielles est

$$(n-1)\ I^2 \quad (I \text{ opérations } * + (I - 1) \text{ réunions})$$

Remarque 2.

Chemins d'un sous-graphe X_K à tout point du graphe, dont le nombre d'arcs est compris entre n_1 et n, et en particulier chemins de X_K à tout point du graphe dont le nombre d'arcs est inférieur ou égal à n. On recherche successivement les chemins dont le nombre d'arcs est $n_1, n_1+1, \ldots, n$ et on en fait la réunion.

Remarque 3.

Chemins de n arcs ou dont le nombre d'arcs est compris entre n_1 et n de tout point du graphe à un sous-graphe X_K. On développera un produit de la forme

$$M \times \ldots M \times M(., K) = M^{n-1}\ M(., K).$$

On pourra pour ce faire transposer tout ce qui a été dit précédemment.

On peut aussi rechercher les chemins d'un sous-graphe à un sous-graphe.

Remarque 4.

Une autre interprétation de l'algorithme est possible. On peut considérer les opérations . comme un ensemble d'opérations unaires . de cardinal égal au nombre des arcs a du graphe :
a

$$\cdot_a \mu = \mu \cdot_i a$$

si μ est un chemin et si i caractérise le sommet origine de l'arc a.

On peut aussi considérer les $\cdot_a$ comme des opérations reinstreintes à des couples ordonnés de chemins et d'arcs.

b. La détermination des chemins de n arcs de tout point à tout point d'un graphe peut être ainsi faite :

Algorithme b.

On calcule successivement M^2, M^4, M^8, ..., M^{2^p}, $M^{2^p+2^q}$, ..., M^n.

A la $i^{ème}$ itération, on obtient une suite sans répétition de chemins ayant le même nombre d'arcs joignant tout point du graphe à tout point du graphe.

Remarque 5.

Le nombre d'opérations au niveau des cases matricielles est

$$NI^3 \text{ opérations } \cdot \text{ et } NI^2 (I - 1) \text{ réunions}$$

où N est le nombre d'itérations.

Remarque 6.

Enumération des chemins de tout point à tout point du graphe, dont le nombre d'arcs est compris entre n_1 et n. On recherchera d'abord les chemins dont le nombre d'arcs est n_1 à l'aide du dernier algorithme (b), puis ceux dont le nombre d'arcs est $n_1 + 1$ soit à partir des chemins de longueur n_1 à l'aide de l'algorithme a, soit à l'aide de l'algorithme b ...

c. Comparaison des algorithmes. Extension.

L'algorithme b est plus rapide que l'algorithme a, en comparant les comptages au niveau des cases matricielles, dès que

$$N < n-1$$

On peut panacher ces algorithmes, rechercher par exemple les chemins de n arcs d'un sous-graphe à tout point du graphe en utilisant d'abord l'algorithme b pour développer un produit de M, puis l'algorithme a pour le multiplier par M (K, .) et les matrices M restantes :

$$\underbrace{M(K,.) \times M \times \ldots \times M \times \underbrace{M \times \ldots \times M}_{\text{algorithme b}}}_{\text{algorithme a}}$$

Ces algorithmes sont utilisables pour tout développement de produit associatif de matrices identiques (l'associativité n'est pas même nécessaire pour l'algorithme a), donc en particulier pour tous les problèmes relatifs aux graphes, étudiés en 6-II-2 et 6-III-3 (voir par exemple [PAIR, DERNIAME], [KAUFMANN 2], [MAGHOUT]). Nous les appliquerons aussi en IV à l'énumération des ensembles couplants maximaux d'un graphe simple.

II.2. Enumération des chemins hamiltoniens, circuits hamiltoniens, facteurs, dissections d'un graphe [BOUCHET].

Par exemple, l'énumération des chemins hamiltoniens du graphe de 6-II-2c s'obtient en développant le produit des ensembles d'arcs incidents à I-1 sommets (les sommets ici choisis sont successivement A, B, C et D) :

(AB+AC+AE+EA) (AB+BC+BE) (AC+BC+CD+DC) (CD+DC+DE+ED)

(ABC+ABE+AC.BE+AE.BC+EAB+EA.BC+BEA)

(ABCD+ABE.CD+ABE.DC+ACD.BE+AE.BCD+EABC+EAB.CD+EAB.DC+EA.BCD+BEAC+BEA.CD+BEA.DC)

ABCDE+ABEDC+EABCD+DEABC+CDEAB+BCDEA+BEACD

Ce procédé a l'intérêt de prendre un produit de facteurs très strict, dont le développement va donner uniquement et sans redondance les chemins hamiltoniens (cela rend en particulier inutile une sélection à la fin du développement), mais a l'inconvénient de considérer des suites de chemins compatibles.

Le développement effectué est progressif; on aurait pu faire un développement différentié dans le monoïde des chemins compatibles.

III - ENUMERATION DES MONOMES PREMIERS D'UNE FONCTION BOOLEENNE

En 3 et en 5.I , nous avons donné des algorithmes permettant d'énumérer les monômes premiers d'une fonction booléenne ; donnons-en maintenant un autre interprétable.

- soit comme le développement progressif d'au plus 2^I facteurs identiques (à la forme canonique de la fonction booléenne);
- soit comme l'algorithme F2 appliqué à un nouveau triplet $(A, \leq, *)$: $*$ sera plus rapide que l'opération consensus (son efficacité proviendra de ce qu'il ne sera défini que sur des couples de monômes ayant le même nombre de lettres), mais l'ensemble de départ A sera moins arbitraire.

Explicitons cet algorithme avec la terminologie de la seconde interprétation.

III.1. Définition de $(A, *, \leq)$.

1. $*$ est maintenant l'opération de consensus reinstreinte aux couples de monômes ayant le même nombre de lettres et les mêmes lettres ; $*$ s'applique aux monômes différents par la complémentation d'une et d'une seule lettre, c'est-à-dire de la forme Px et Px' : $Px * Px' = P$, et donne pour $*$ - composé de deux monômes de i lettres un monôme de i-1 lettres. Alors :

2. A est maintenant la forme canonique A_I de la fonction booléenne f (ou ensemble des monômes de I lettres de f) ou l'ensemble A_{i_o} des monômes premiers à plus de i_o lettres et des monômes de f à i_o lettres ($i_o = I-1, I-2, \ldots, 1$) ;

3. $\leq$ est maintenant caractérisé par : deux monômes de f sont incomparables.

III.2. Algorithme.

F_2 donne :

Algorithme.

1. On part de A_i écrit en file, les premiers éléments de la file étant les monômes premiers à plus de i lettres, les éléments suivants étant tous les monômes à i lettres de f classés

- d'abord en ensembles $A_{i,j}$ de monômes ayant les mêmes lettres (caractérisés par j) ;
  ensuite, dans chacun de ces ensembles $A_{i,j}$ (j variable), par ordre décroissant du nombre k de lettres accentuées ($0 \leq k \leq i$) ;
  appelons $A_{i,j,k}$ les classes de monômes obtenues (j,k variables).

2. Pour chaque $A_{i,j}$ (j variable), on recherche l'$*$ - composé de chaque élément d'un $A_{i,j,k}$ (en commençant par le deuxième $A_{i,j,k}$ à partir de la gauche s'il existe, en terminant par le $A_{i,j,k}$ en queue) avec chaque élément de $A_{i,j,k+1}$ s'il existe ; les $*$ - composés trouvés appartiennent à un $A_{i-1,\ell,k}$ (ℓ à déterminer) ; ils sont ainsi écrits : dès qu'un $*$ - composé est formé, on coche les éléments qui lui ont donné naissance, on l'ajoute en fin de liste s'il n'est pas un élément déjà trouvé de $A_{i-1,\ell}$; sinon on l'élimine.

A_{i-1} est l'ensemble des éléments non cochés.

3. On itère à partir de $A = A_{i_o}$ 1) et 2) jusqu'à ce qu'il n'y ait plus de création de monômes, Les éléments non cochés sont les monômes premiers.

Remarque 1.

Cet algorithme est celui de McCLUSKEY [KUNTZMANN] p. 109 amélioré par [HARRISON] p. 116 . Pour éviter à chaque étape le classement par monômes ayant les mêmes lettres, on peut utiliser la méthode suivante inspirée de la condition C : tout couple d'éléments extraits respectivement de $A_{i,..,k}$ et $A_{i,..,k+1}$ a un $\overset{*}{}_h$ - composé, si, abstraction faite de leurs $h^{èmes}$ bits, les éléments sont identiques ; on itère sur h.

Remarque 2.

Cocher ne fait pas partie de l'algorithme F2 de recherche des éléments maximaux ; c'est un moyen commode de sélectionner les monômes premiers au fur et à mesure de leur formation, évitant un tri final.

Remarque 3.

F1 donne le même algorithme à condition de prendre dans chacune de ses étapes, l'ordre de F2 ; sinon, l'algorithme obtenu est très semblable.

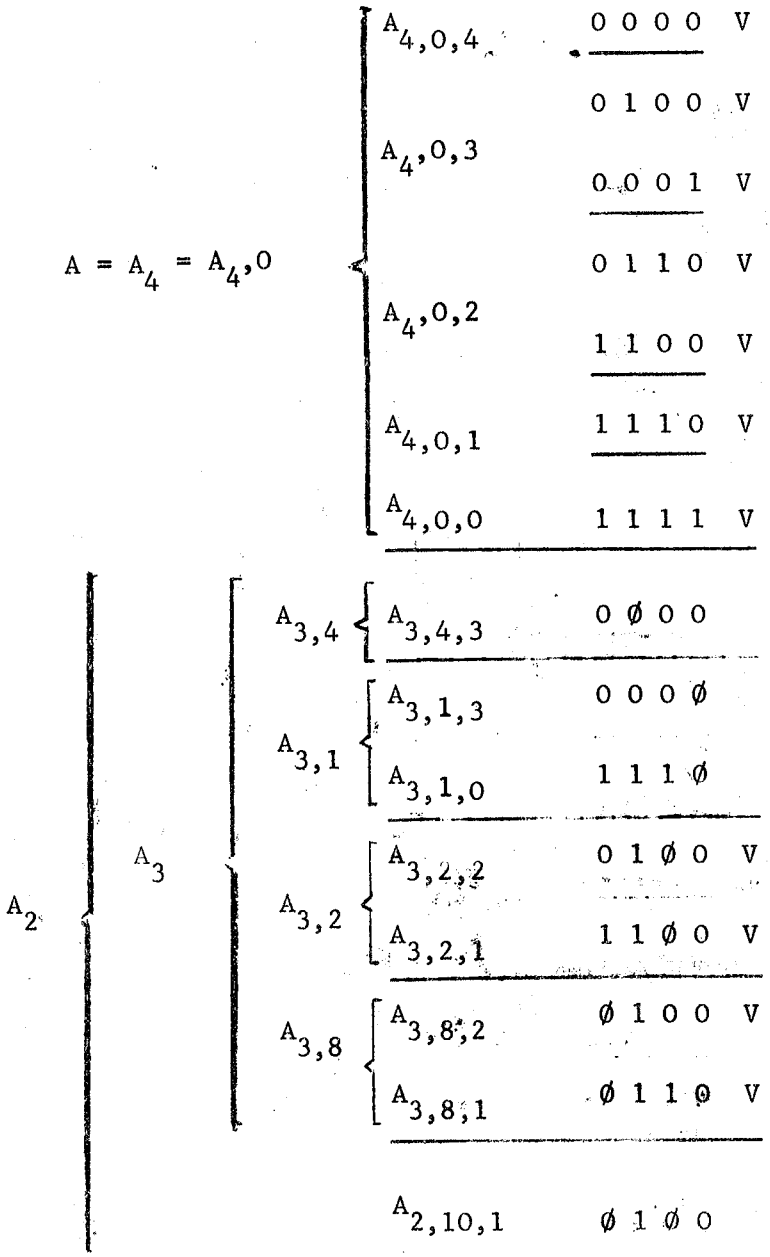
Remarque 4.

La condition (C) n'est pas vérifiée.

Remarque 5.

La méthode est généralisable à la recherche des pavés maximaux dans un produit de treillis distributifs.

III.3. **Exemple de [HARRISON] p. 116.**



IV - ENUMERATION DES ENSEMBLES COUPLANTS MAXIMAUX D'UN GRAPHE SIMPLE

Etant donné un graphe simple (Y, Z, Γ) et des sous-ensembles B de Y et C de Z tels que Card B = Card C, rappelons qu'un couplage de B dans Z ou de B sur C est toute bijection γ de B sur C telle que

$$\forall\, y_j \in B : \gamma\, y_j \subseteq \Gamma\, y_i$$

(B,C) est appelé ensemble couplant de dimension Card B ; ils sont dits maximaux lorsque le graphe simple n'admet pas de couplage dont les ensembles couplants contiennent strictement B et C [ORE] p. 132.

IV.1. Choix du triplet (A, $*$, $\leq$).

1. Les éléments de G sont les ensembles couplants (B, C) ; ils vérifient $B \subseteq Y$, $C \subseteq Z$, Card B = Card C.

2. La relation d'ordre dans G est définie par

$(B, C) \leq (D, E) \Leftrightarrow B \subseteq D$ et $C \subseteq E$.

3. $*$ est l'opération binaire

$(B, C) * (D, E) = (B \cup D,\ C \cup E)$

si $B \cap D = \emptyset$ et $C \cap E = \emptyset$; sinon, elle n'est pas définie.

En effet, la connaissance de deux ensembles couplants non disjoints (B, C) et (D, E) ($B \cap D \neq \emptyset$ ou $C \cap E \neq \emptyset$) ne permet pas de définir un nouvel ensemble couplant : par exemple, $(y_1\ y_3, z_1\ z_3)$ et $(y_2 y_3, z_2\ z_3)$

sont des ensembles couplants d'un des graphes simples suivants :

	z_1	z_2	z_3
y_1	o		
y_2			□
y_3		□	o

	z_1	z_2	z_3
y_1	o		
y_2		□	
y_3			□

	z_1	z_2	z_3
y_1			o
y_2			□
y_3	o	□	

	z_1	z_2	z_3
y_1			o
y_2		□	
y_3	o		□

Pour déterminer le graphe simple, la connaissance d'autres ensembles couplants ou la définition d'une opération $*$ d'ordre supérieur à 2 est donc nécessaire.

4. L'ensemble de départ A est l'ensemble des couples (y_j, z_k) (attachés à chaque arête d'extrémités $y_j \in Y$ et $z_k \in Z$).

IV.2. Algorithmes

Avec le triplet défini ci-dessus, on peut construire facilement plusieurs algorithmes puisque les ensembles couplants maximaux ont tous même cardinalité.

1. A partir des ensembles couplants de dimension i, on construit les ensembles couplants de dimension i + 1, en utilisant les ensembles couplants de dimension 1 ou arêtes. Ceci revient à définir une opération $*_{y_j z_k}$ unaire à partir de chaque arête $y_j z_k$ du graphe de la façon suivante :

$$\begin{cases} *_{y_j z_k}(B, C) = (B \cup y_j , C \cup z_k) & \text{si } B \cap y_j = C \cap z_k = \emptyset \\ *_{y_j z_k}(B, C) \text{ non défini} & \text{sinon} \end{cases}$$

2. A partir des ensembles couplants de dimension i, on construit les ensembles couplants de dimension 2i s'ils existent, sinon ceux de dimension $i + \frac{i}{2}$ ...

3. On peut aussi, à partir d'une arête $y_j z_k$ lui appliquer les opérateurs $*$ de toutes les façons possibles, ou, de façon équivalente, faire un développement différentié après avoir ordonné totalement les arêtes :

$$(y_1,z_1) + (y_1,z_2) + (y_2,z_1) + (y_2,z_3) + (y_3,z_2) + (y_3,z_3) + (y_1,z_3)$$

Il est inutile d'aller plus loin puisqu'il existe un seul ensemble couplant maximal $(\{y_1, y_2, y_3\}, \{z_1,z_2,z_3\})$.

V - PAVES MAXIMAUX DANS UN PRODUIT DE TREILLIS DISTRIBUTIFS ET DE TREILLIS BOOLEENS

Donnons un autre algorithme que ceux de 3 , de 5-I et de 7-III pour trouver les pavés maximaux compatibles avec une partie finie du produit G de I treillis distributifs , algorithme qui est une extension de celui de recherche des monômes premiers d'une fonction booléenne par double dualisation [KUNTZMANN 2].

V.1. Préliminaires.

Les notations sont celles de 3-I-1 et 3-I-2. Cependant, au lieu de considérer dans l'ensemble G des pavés les opérations $\underset{i}{*}$ $(i = 1, 2, \ldots, I)$,

nous considérerons maintenant la seule opération produit ou borne inférieure (voir 3-I-2d) notée . , définie de la façon suivante :

$$(b_1,b_2,\ldots,b_I) \,.\, (c_1,c_2,\ldots,c_I) = (b_1 \underset{1}{.} c_1, b_2 \underset{2}{.} c_2, \ldots, b_I \underset{I}{.} c_I)$$

Rappelons que l'ordre sur G est le produit des ordres sur les G_i ; il vérifie, pour tous pavés b et c :

$$b \leq c \quad <=> \quad b \,.\, c = b$$

puisque

$$b \leq c \;\; <=> \;\; b_i \underset{i}{\leq} c_i \;\; (i=1,2,\ldots,I) <=> b_i \underset{i}{.} c_i = b_i \;\; (i=1,2,\ldots,I) <=>$$

$$bc = b$$

Il en résulte l'isotonie du produit, ou, pour tout pavé d :

$$b \leq c \Rightarrow b.d \leq c.d \qquad (I)$$

Dans ces conditions, en introduisant comme dans 4-I-1b l'équivalence $\equiv$ dans l'ensemble des parties de G ($B \equiv C \Leftrightarrow B \leq C$ et $C \leq B$, où $\leq$ est l'extension suivante à $\mathcal{P}(G)$ de l'ordre $\leq$ de G : $B \leq C \Leftrightarrow$ pour tout élément b de B, il existe un élément c de C tel que $b \leq c$), on peut parler des lois de composition interne réunion et produit dans l'ensemble quotient $\mathcal{P}(G)/\equiv$ ou dans l'ensemble des éléments Max A de $\mathcal{P}(G)$.
Ces lois structurent $\mathcal{P}(G)/\equiv$ en un treillis distributif puisque

$\cup$ et . sont associatives, commutatives, idempotentes

$B \,.\, (B \cup C) \equiv B$, $B \cup (B \,.\, C) \equiv B$

est distributif par rapport à $\cup$: $B \,.\, (C \cup D) = (B \,.\, C) \cup (B \,.\, D)$

(donc $B \cup (C \,.\, D) \equiv (B \cup C) \,.\, (B \cup D)$). Soit O (respectivement 1) le produit des plus petits éléments O_i (respectivement des plus grands éléments 1_i) des sous-treillis des G_i engendrés par les $i^{ème}$ composantes des pavés de A.

Nous écrirons un pavé $(b_1, b_2, \ldots, b_I)$: $b_1 \,.\, b_2 \,.\, \ldots \,.\, b_I$ en faisant la convention :

$$b_i = (1_1, \ldots, 1_{i-1}, b_i, 1_{i+1}, \ldots, 1_I) \,.$$

V.2. Recherche de pavés maximaux par double dualisation.

a. Théorème.

Les pavés maximaux du produit de deux parties de G sont obtenus en développant le produit des pavés maximaux des deux parties, puis en réduisant.

Démonstration .

Tout pavé du produit de deux ensembles est, d'après (I), inférieur ou égal au produit de leurs pavés maximaux.
Inversement, le produit d'un pavé de chaque ensemble est évidemment un pavé de leur produit.

Autre démonstration

Elle fait appel aux opérations $\overset{*}{i}$ introduites en 3-I-1.

Montrons qu'après avoir effectué le produit des pavés maximaux des deux parties, aucune opération $\overset{*}{i}$ ne peut donner de nouveaux pavés maximaux.

En effet, deux pavés arbitraires b. d et c . e du produit

$$(b \cup c \cup \ldots) \,.\, (d \cup e \cup \ldots)$$

ont pour $\overset{*}{i}$ - composé

$$(b.d) \overset{*}{i} (c.e) = (\ldots, b_{i-1} \underset{i-1}{.} d_{i-1}, \underset{i-1}{.} c_{i-1} \underset{i-1}{.} e_{i-1},$$

$$(b_i \underset{i}{.} d_i) \overset{+}{i} (c_i \underset{i}{.} e_i), b_{i+1} \underset{i+1}{.} d_{i+1} \underset{i+1}{.} c_{i+1} \underset{i+1}{.} e_{i+1}, \ldots)$$

$$\leq (\ldots, b_{i-1} \underset{i-1}{.} c_{i-1}, b_i \overset{+}{i} c_i, b_{i+1} \underset{i+1}{.} c_{i+1}, \ldots) \,.$$

$$(\ldots, d_{i-1} \underset{i-1}{.} e_{i-1}, d_i \overset{+}{i} e_i, d_{i+1} \underset{i+1}{.} e_{i+1}, \ldots)$$

$$= (b \overset{*}{i} c) \,.\, (d \overset{*}{i} e).$$

$b \overset{*}{i} c$ et $d \overset{*}{i} e$ étant inférieurs ou égaux à un pavé des facteurs du produit, la condition (I) entraîne que $(b \overset{*}{i} c) \,.\, (d \overset{*}{i} e)$ (donc $(b \,.\, d) \overset{*}{i} (c \,.\, e)$) est inférieur ou égal à au moins un pavé du produit.

b. Application à la recherche de pavés maximaux par double dualisation.

A partir d'une partie A de $\mathcal{P}(G)$ écrite comme une réunion de pavés, on obtient sa duale A^d par échange des opérations . et $\cup$ et des éléments zéro et unité ; A^d est alors donné par un produit de crochets, chaque crochet étant une réunion de pavés (maximaux relativement aux crochets) ; en développant le produit et en réduisant, on obtient donc , d'après le théorème précédent, les pavés maximaux de A^d.

Lors du développement du produit, il y a interférence de $\overset{*}{i}$ et de . comme le montre le passage de (1) à (2) dans l'exemple qui suit et l'opérateur dualisation d n'est pas de carré l'identité. Par contre une double dualisation de A obtenue par échange de . et $\cup$, de $\overset{*}{i}$ et $\overset{+}{i}$ fournit ses pavés maximaux.

Exemple.

Considérons les deux rectangles (ou pavés à deux composantes) suivants de $G_1 \times G_2$:

$$A = [a_1 \,.\, (b_2 + c_2)] \cup [b_1 \,.\, (a_2 + c_2)]$$

G_1 et G_2 sont supposés des treillis de Boole ayant pour atomes respectivement a_1, b_1 et a_2, b_2, c_2. Nous pouvons représenter A par le diagramme suivant :

A	a_2	b_2	c_2
a_1		x	x
b_1	x		x

Les rectangles maximaux de A^d sont obtenus en développant :

$$A^d = [a_1 \cup (b_2 +_2 c_2)] \cdot [b_1 \cup (a_2 +_2 c_2)]$$

soit

$$A^d = (a_1 \cdot b_1) \cup [a_1 \cdot (a_2 +_2 c_2)] \cup [b_1 \cdot (b_2 +_2 c_2)] \cup [(b_2 +_2 c_2) \cdot (a_2 +_2 c_2)] \quad (1)$$

$$= (a_1 \cdot_1 b_1) \cup [a_1 \cdot (a_2 +_2 c_2)] \cup [b_1 \cdot (b_2 +_2 c_2)] \cup [(b_2 +_2 c_2) \cdot_2 (a_2 +_2 c_2)] \quad (2)$$

$$= 0_1 \cup [a_1 \cdot (a_2 +_2 c_2)] \cup [b_1 \cdot (b_2 +_2 c_2)] \cup c_2$$

Les rectangles maximaux de A sont eux donnés par:

$$A^* = [a_1 \cup (b_2 \cdot_2 c_2)] \cdot [b_1 \cup (a_2 \cdot_2 c_2)] \quad (3)$$

$$= (a_1 \cdot b_1) \cup [a_1 \cdot (a_2 \cdot_2 c_2)] \cup [b_1 \cdot (b_2 \cdot_2 c_2)] \cup [(b_2 \cdot_2 c_2) \cdot (a_2 \cdot_2 c_2)]$$

$$= (a_1 \cdot_1 b_1) \cup [a_1 \cdot (a_2 \cdot_2 c_2)] \cup [b_1 \cdot (b_2 \cdot_2 c_2)] \cup [(b_2 \cdot_2 c_2) \cdot_2 (a_2 \cdot_2 c_2)]$$

$$= 0_1 \cup [a_1 \cdot (a_2 \cdot_2 c_2)] \cup [b_1 \cdot (b_2 \cdot_2 c_2)] \cup 0_2$$

$$A = A^{**} = 1_1 \cdot [a_1 \cup (a_2 +_2 c_2)] \cdot [b_1 \cup (b_2 +_2 c_2)] \cdot 1_2 \qquad (4)$$

Or $1_1 = 1_2 = 1$; d'où

$$A^{**} = (a_1 \cdot b_1) \cup [a_1 \cdot (b_2 +_2 c_2)] \cup [b_1 \cdot (a_2 +_2 c_2)] \cup [(a_2 +_2 c_2) \cdot (b_2 +_2 c_2)]$$

$$= 0_1 \cup [a_1 \cdot (b_2 +_2 c_2)] \cup [b_1 \cdot (a_2 +_2 c_2)] \cup c_2$$

Les rectangles maximaux à composantes non nulles compatibles avec A sont donc :

$$\boxed{[a_1 \cdot (b_2 +_2 c_2)] \cup [b_1 \cdot (a_2 +_2 c_2)] \cup c_2}$$

Remarque 1.

$b_2 \cdot_2 c_2$ et $a_2 \cdot_2 c_2$ sont dans (3) des expressions formelles ; il serait absurde de les remplacer par 0_2 puisque $b_2 +_2 c_2$ et $a_2 +_2 c_2$ sont différents de 1_2 .

Remarque 2.

La double dualisation est une façon de formaliser le processus conduisant à la détermination des éléments maximaux. Une autre méthode est d'opérer directement à partir de A réunion de pavés en utilisant la distributivité de la réunion par rapport au produit :

$$A = (a_1 \cup b_1) \cdot [a_1 \cup (a_2 +_2 c_2)] \cdot [b_1 \cup (b_2 +_2 c_2)] \cdot [(b_2 +_2 c_2) \cup (a_2 +_2 c_2)]$$

Pour que le théorème soit applicable, chaque crochet doit être la réunion de pavés maximaux ; A doit donc être écrit :

$$A = (a_1 +_1 b_1) \cdot [a_1 \cup (a_2 +_2 c_2)] \cdot [b_1 \cup (b_2 +_2 c_2)] \cdot [(b_2 +_2 c_2) +_2 (a_2 +_2 c_2)] \quad (4)$$

$$A = [a_1 \cup (a_2 +_2 c_2)] \cdot [b_1 \cup (b_2 +_2 c_2)] \cdot$$

Il ne reste plus qu'à développer pour obtenir les rectangles maximaux de A.

Remarque 3

Avant de dualiser A et A^*, on peut supprimer les pavés ayant une composante nulle et, pour A^* si les seuls pavés à composantes non nulles sont recherchés, les composantes égales à l'unité. Cette remarque sera utilisée en VI-1.

V-3. Recherche de pavés maximaux dans un produit de treillis booléens par complémentation.

a. Cas général.

On peut obtenir une partie A de G à partir de son complément A' écrite comme une réunion de pavés, en échangeant les opérations produit et réunion et en changeant les éléments de G_i en leurs compléments ; le théorème 2a assure que le développement et la réduction de l'expression obtenue de A fournit les pavés maximaux de A.

Reprenons l'exemple précédent. Les rectangles maximaux de A' sont obtenus en développant :

$$A' = [b_1 \cup a_2] \cdot [a_1 \cup b_2]$$

soit

$$A' = [a_1 \,.\, a_2] \cup [b_1 \,.\, b_2]$$

Les rectangles maximaux de A = A'' sont alors donnés par

$$A = [b_1 \cup (b_2 \underset{2}{+} c_2)] \,.\, [a_1 \cup (a_2 \underset{2}{+} c_2)] \qquad (5)$$

$$\boxed{A = [a_1 \,.\, (b_2 \underset{2}{+} c_2)] \cup [b_1 \,.\, (a_2 \underset{2}{+} c_2)] \cup c_2}$$

Remarque 1.

Il est évidemment intéressant de trouver les pavés maximaux de A par un seul développement ; pour ce faire, on détermine A' et on complémente.

En particulier, si A' est de volume faible par rapport à A ou si, quand les pavés sont des rectangles, le diagramme représentatif de A est connu, on peut d'abord aisément déterminer les pavés atomiques de A', puis développer le produit des compléments (considérés comme la réunion de I pavés maximaux) de chacun de ces pavés ; il se trouve que (5) , dans l'exemple précédent, est un tel produit , les rectangles maximaux de A' étant des rectangles atomiques.

Remarque 2.

Dans le cas où sont cherchés les seuls pavés symétriques (ou pavés dont les composantes sont toutes identiques) maximaux par développement de

$$(b_1, b_2, \ldots, b_I) \,.\, (c_1, c_2, \ldots, c_I) \,.\, \ldots \qquad (I)$$

il suffit de développer, en adoptant la notation supposée possible

$\inf_i b_i = b_1 \,.\, b_2 \,.\, \ldots \,.\, b_I$ $\quad$ (j,k,...,ℓ arbitraires) :

$$(\inf_i b_i,\ \inf_i b_i,\ \ldots,\ \inf_i b_i) \,.\, (\inf_i c_i,\ \inf_i c_i,\ \ldots,\ \inf_i c_i) \,.\, \ldots$$

ou encore plus simplement, les composantes des pavés symétriques maximaux sont le résultat du développement de

$$(\inf_i b_i) \,.\, (\inf_i c_i) \,.\, \ldots$$

b. Application à la détermination des ensembles fortement stables et des cliques (ou sous-ensembles de sommets deux à deux adjacents) maximaux d'un hypergraphe.

Prenons comme exemple le graphe (sans boucles) Γ de la page 42 de [BERGE 1]; considérons la matrice booléenne symétrique à diagonale vide M associée au graphe non orienté Γ^* associé à Γ (la matrice booléenne symétrique ici associée à un hypergraphe est la matrice à lignes et à colonnes caractérisées par les sommets de l'hypergraphe et à élément égal à 1 si et seulement si les sommets correspondants sont distincts et adjacents) :

M	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1		1	1				
x_2	1			1	1	1	
x_3	1				1		
x_4		1			1		
x_5		1	1	1			1
x_6		1					1
x_7					1	1	

et cherchons les sous-ensembles intérieurement stables maximaux de Γ (ou Γ') ou cliques maximales du graphe non orienté $(\Gamma')'$ complémentaire de Γ' [BERGE 1], c'est à dire les composantes des carrés nuls (vides sur la figure ci-jointe) symétriques maximaux de M. Pour cela, nous utilisons la première remarque en formant le produit des compléments des carrés atomiques de M

$$|(x_1',1) \cup (1,x_2')| \cdot |(x_1',1) \cup (1,x_3')| \cdot |(x_2',1) \cup (1,x_4')| \cdot \ldots \qquad (1)$$

où

$$1 = x_1 \underset{i}{+} x_2 \underset{i}{+} x_3 \underset{i}{+} x_4 \underset{i}{+} x_5 \underset{i}{+} x_6 \underset{i}{+} x_7,$$

$$x_1' = x_2 \underset{i}{+} x_3 \underset{i}{+} x_4 \underset{i}{+} x_5 \underset{i}{+} x_6 \underset{i}{+} x_7,$$

$$x_2' = x_1 \underset{i}{+} x_3 \underset{i}{+} x_4 \underset{i}{+} x_5 \underset{i}{+} x_6 \underset{i}{+} x_7, \ \ldots$$

i valant 1 ou 2 selon le cas, et la seconde remarque en formant le produit de

$$(x_1' \cup x_2') \cdot (x_1' \cup x_3') \cdot (x_2' \cup x_4') \cdot (x_2' \cup x_5') \cdot (x_2' \cup x_6') \cdot (x_3' \cup x_5') \cdot$$
$$(x_4' \cup x_5') \cdot (x_5' \cup x_7') \cdot (x_6' \cup x_7'). \qquad (2)$$

Nous obtenons :

$$(x_1' \cdot x_2' \cdot x_5' \cdot x_6') \cup (x_1' \cdot x_2' \cdot x_5' \cdot x_7') \cup (x_1' \cdot x_4' \cdot x_5' \cdot x_6') \cup (x_2' \cdot x_3' \cdot x_4' \cdot x_7') \cup (x_2' \cdot x_3' \cdot x_5' \cdot x_6') \cup$$
$$(x_2' \cdot x_3' \cdot x_5' \cdot x_7')$$

c'est à dire :

$\{x_3,x_4,x_7\}$, $\{x_3,x_4,x_6\}$, $\{x_2,x_3,x_7\}$, $\{x_1,x_5,x_6\}$, $\{x_1,x_4,x_7\}$ et $\{x_1,x_4,x_6\}$

On reconnait les résultats de [MAGHOUT] ou de [BENZAKEN 1] relatifs aux sous-ensembles intérieurement stables maximaux d'un graphe ; on vérifie aussi que c'est algorithme de [BEDNAREK, TAULBEE] à condition de développer (2) de la façon suivante :

$$(x'_2 \cup x'_1) \cdot (x'_3 \cup x'_1) \cdot (x'_4 \cup x'_2) \cdot (x'_5 \cup x'_2\, x'_3\, x'_4) \cdot (x'_6 \cup x'_2) \cdot (x'_7 \cup x'_5\, x'_6)$$

c'est-à-dire, en calculant successivement les sous-ensembles intérieurement stables maximaux des sous-graphes du graphe donné de sommets $\{x_1,x_2\}$, $\{x_1,x_2,x_3\}$, ..., $\{x_1,x_2,x_3,x_4,x_5,x_6\}$ et $\{x_1,x_2,x_3,x_4,x_5,x_6,x_7\}$ ou encore en développant d'abord les facteurs ne contenant pas x'_3, x'_4,..., x'_7, puis ceux ne contenant pas x'_4, x'_5,..., x'_7, ..., puis ceux ne contenant pas x'_7, enfin ceux contenant x'_7 . et à condition de développer l'intersection de deux facteurs suivant des règles précises ; d'ailleurs, en VI-3-c, sera donné un algorithme de la même veine mais meilleur.

Cette méthode de recherche des cliques et ensembles fortement stables sera retrouvée en VI-4 ; elle se réduit au développement d'un produit de réunions de deux lettres, développements qui sont systèmatiquement étudiés dans le paragraphe suivant ; seulement la manière dont vient d'être ici introduite cette méthode est peut être plus parlante.

VI - ALGORITHMES DE COUVERTURE MINIMALE.

VI.1. Définition d'une couverture minimale.

Les notations sont celles de V-1, mais nous nous limitons au cas d'un produit $G = G_1 \times G_2 \times \ldots \times G_I$ de treillis distributifs G_i ayant, en plus des plus petit et plus grand élément, un seul élément $x_i (0_i < x_i < 1_i)$. Le problème envisagé en V-2b du développement d'un produit de réunions de pavés maximaux devient un problème de <u>développement d'un produit de sommes de lettres</u> x_i (à condition de ne considérer que les éléments du développement n'ayant aucune composante nulle) :

$$(x_{1_1} + x_{1_2} + \ldots + x_{1_M}) \cdot (x_{2_1} + x_{2_2} + \ldots + x_{2_Q}) \ldots . (x_{n_1} + x_{n_2} + \ldots + x_{n_R})$$

La dénomination de somme et la notation + remplaceront respectivement réunion et $\cup$ (voir 3-III-1a)
Nous appellerons aussi ce problème <u>développer un produit de facteurs</u> ; il consiste à transformer le produit de sommes en somme de produits et à simplifier cette somme, en utilisant l'associativité, la commutativité et l'idempotence du produit, en supprimant les éléments (ou monômes) multiples d'un autre.

Ce problème peut être interprété comme celui de la détermination des <u>couvertures minimales</u> de n points (ou ensembles minimaux en nombre de parties couvrant les n points) sachant qu'ils sont couverts respectivement par les parties :

$$x_{1_1}, x_{1_2}, \ldots, x_{1_M}$$
$$x_{2_1}, x_{2_2}, \ldots, x_{2_Q}$$
$$\ldots$$
$$x_{n_1}, x_{n_2}, \ldots, x_{n_R}$$

Avant de, à proprement parler, développer, on peut d'abord simplifier chacun des facteurs (si des lettres sont répétées), puis l'ensemble des facteurs. On peut aussi rechercher les symétries entre les lettres et essayer de décomposer le produit de facteurs, c'est-à-dire trouver un partitionnement des facteurs qui partitionne les lettres (on utilisera pour cela l'algorithme de 5-II-2b) ; on développera alors les facteurs de chacune des classes séparément (avec les algorithmes qui vont être exposés), puis on développera les quantités obtenues sans faire de réduction car elles sont inutiles.

Par exemple, pour $(a + b + c)(d + e)$, les ensembles $\{a, b, c\}$ et $\{d, e\}$ étant disjoints, les données immédiates

$$ad + bd + cd + ae + be + ce$$

sont les couvertures minimales. Pour $A = (a + b + c)(d + e)(d + f)(e + f)$, le fait que $\{a, b, c\}$ et $\{d, e, f\}$ sont disjoints permet de développer indépendamment $A_1 = a + b + c$ et $A_2 = (d + e)(d + f)(e + f) = de + df + ef$, puis $A_1 A_2$ sans chercher à réduire :

$$A = (a + b + c)(de + df + ef)$$
$$A = ade + adf + aef + bde + bdf + bef + cde + cdf + cef$$

Nous supposns dorénavant que les facteurs ont au moins deux lettres.

Les deux catégories de développement explicitées en I-3 donnent, en remarquant que :

- les opérations o et * sont maintenant, en termes d'algèbre de Boole, le produit et, en termes de théorie des ensembles, la réunion,
- la relation d'ordre est la relation entre monômes booléens ou la relation duale de l'inclusion ensembliste,

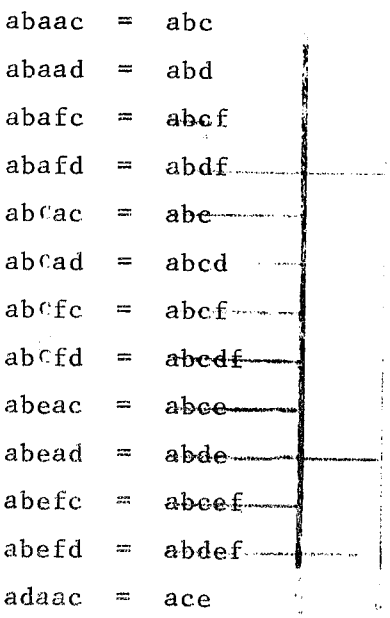
(o et * sont donc isotones), les deux groupes suivants d'algorithmes de couverture minimale (ou détermination des monômes premiers de produits de sommes).

VI.2. Algorithmes différentiés lexicographiques de couverture minimale.

L'application brutale du développement différentié de I-3a à l'exemple suivant :

$$(a + b + d)(b + d + e + f)(a + c + e)(a + f)(c + d)$$

donne, en prenant un terme dans chacun des facteurs de toutes les façons possibles, en utilisant l'idempotence et en réduisant :



Au lieu d'énumérer les n-uplets et d'en déduire les produits, on peut grouper ces deux phases en utilisant un ordre total arbitraire sur les termes des facteurs, appelé ordre lexicographique.

a. Algorithme lexicographique direct de couverture minimale.

Algorithme.

On construit toutes les sous-suites lexicographiques minimales (ayant le minimum de lettres) contenant au moins une lettre de chaque facteur de la façon suivante :

On élimine les facteurs contenant la lettre introduite dans la sous-suite lexicographique en formation ; quand il n'y a plus de facteurs ou plus de possibilité d'introduction de nouvelles lettres, on construit la sous-suite lexicographique suivante si elle existe.

Toute suite dont l'une des lettres n'a pas entraîné de suppression de facteurs ou telles qu'il existe encore des facteurs à la fin de sa formation (la dernière lettre est alors nécessairement la dernière lettre lexicographique) est éliminée. Les autres, après réduction, sont les éléments du développement.

Cet algorithme donne sur l'exemple précédent :

abc
abd
aed
ace
acf
ad
bcf
bdef
cdf
def

Remarque.

On peut programmer l'algorithme précédent en numérotant les facteurs du développement et en associant à chaque lettre les numéros des facteurs la contenant. L'algorithme devient :

On forme, dans l'ordre lexicographique des lettres, des suites et des suites associées de numéros de la façon suivante : quand une lettre est introduite dans la suite en formation, les numéros associés à la lettre sont introduits dans la suite associée. Quand on ne peut introduire de nouvelle lettre dans la suite en formation ou de nouveaux numéros dans sa suite associée, on construit la suite suivante si elle existe. Toute suite dont la suite associée n'est pas l'ensemble des numéros ou dont une lettre a été introduite sans entraîner l'introduction de nouveaux numéros associés est supprim ; les autres, après réduction, sont les éléments du développement.

Ceci donne sur l'exemple précédent :

$$(a + b + d)(b + e + f + d)(c + a + e)(a + f)(c + d)$$
$$\quad 1 \qquad\quad 2 \qquad\quad 3 \qquad 4 \qquad 5$$

$$a^{134}\ ,\ b^{12}\ ,\ c^{35}\ ,\ d^{125}\ ,\ e^{23}\ ,\ f^{24}$$

$$a^{134}\, b^{2}\, c^{5} + a^{134}\, b^{2}\, d^{5} + a^{134}\, c^{5}\, d^{2} + a^{134}\, c^{5}\, f^{2} + a^{134}\, d^{25}$$
$$+ b^{12}\, c^{35}\, f^{4} + b^{12}\, d^{5}\, e^{3}\, f^{4} + c^{35}\, d^{12}\, f^{4} + d^{125}\, e^{3}\, f^{4}$$

C' est, à une astuce d'écriture près, l'algorithme de [KUNTZMANN 1] p. 106 pour rechercher les sous-ensembles extérieurement stables maximaux d'un graphe.

b. Algorithme lexicographique indirect de couverture minimale.

Cet algorithme aura la propriété très intéressante de ne pas nécessiter de réduction, les monômes premiers du développement étant obtenus une

fois et une seule et les monômes non premiers étant supprimés indépendamment des autres monômes. Donc même s'il est arrêté en cours de route, on sera assuré que les monômes obtenus sont premiers et distincts.

1. Cas général.

Algorithme.

On construit, dans l'ordre lexicographique des lettres, des suites appelées suites interdites ou suites de lettres interdites, et les suites nécessaires correspondantes (ou suites de lettres nécessaires, distinctes des lettres interdites) de la façon suivante : on élimine du produit de facteurs la lettre nouvellement introduite dans la suite interdite en formation, puis on élimine les facteurs à une seule lettre soit x_ℓ ou contenant une telle lettre x_ℓ, enfin on ajoute x_ℓ à la suite nécessaire en formation.

Quand il n'existe plus de lettre pouvant être introduite dans la suite interdite, on forme la suite interdite qui la suit lexicographiquement, si elle existe, ainsi que la suite nécessaire associée.

Le développement des facteurs est l'ensemble des suites nécessaires telles que toute lettre n'y appartenant pas appartient à la suite interdite correspondante, sans omission, ni répétition.

Démonstration.

En effet, en fin de construction d'une suite interdite, les deux cas suivants peuvent se produire :

- ou toute lettre est interdite ou nécessaire : alors la suite nécessaire est une couverture minimale (que toute lettre soit interdite ou nécessaire entraîne qu'il n'y a plus de facteurs, donc que la suite nécessaire est une couverture ; si elle n'était pas minimale,

on pourrait supprimer un de ses termes x_ℓ : c'est impossible puisqu'il existe un facteur contenant uniquement x_ℓ et des termes interdits) et toute couverture minimale est ainsi obtenue une fois et une seule (à cause de la construction lexicographique des suites interdites englobant toute lettre non nécessaire) ;

- ou il existe une lettre qui n'est ni interdite, ni nécessaire : ce cas n'est pas considéré.

Exemple.

$$(a + b + d)(b + d + e + f)(a + c + e)(a + f)(c + d)$$

Les suites nécessaires seront placées en exposant des suites interdites.:

$a^{f}\ b^{d}\ c^{e}$

$a^{f}\ b^{d}\ e^{c}$

$a^{f}\ c^{de}$ puisque b n'est ni lettre interdite, ni lettre nécessaire.

$a^{f}\ d^{bc}\ e$

$a^{f}\ e^{c}$

$bc^{d}\ e^{a}\ f$

$bc^{d}\ f^{a}$

$bd^{ac}\ e^{f}$

$bd^{ac}\ f^{e}$

bef^{da}

bf^{a}

$c^{d}\ e^{a}\ f$

$c^{d}\ f^{a}$

$d^{c}\ ef^{ab}$

$d^{c}\ f^{a}$

$e\ f^{a}$

f^{a}

Les couvertures minimales sont donc : def, cdf, bcf, ad, acf, ace, abc.

Remarque 1.

Pour test d'arrêt dans la construction d'une suite interdite, en sus de l'impossibilité d'adjoindre une nouvelle lettre (qui ne soit pas une lettre nécessaire et qui soit lexicographiquement après la dernière lettre adjointe), on peut prendre la suppression de tous les facteurs du développement. Les deux derniers alinéas de l'algorithme deviennent alors :

Quand il n'existe plus de facteur ou quand il n'existe plus de lettre pouvant être introduite dans la suite interdite, on forme la suite interdite qui la suit lexicographiquement, si elle existe, ainsi que la suite nécessaire associée.

Le développement des facteurs est l'ensemble des suites nécessaires telles que toute lettre n'y appartenant pas et lexicographiquement avant la dernière lettre de la suite interdite correspondante appartient à cette suite interdite, sans omission, ni répétition.

Remarque 2.

D'un point de vue gestion de mémoires, on peut se contenter d'utiliser $n + 3(I-1)$ mémoires, si n est le nombre de facteurs et I le nombre de lettres (les trois tableaux de I-1 mémoires contiennent la dernière lettre interdite, les lettres interdites et les lettres nécessaires). Pour

écourter la recherche des facteurs contenant une seule lettre qui ne soit pas interdite, on peut réordonner les facteurs ne contenant pas de lettres nécessaires en ceux contenant de nouvelles lettres nécessaires placés en tête et en ceux ne contenant toujours pas de lettres nécessaires placés en queue ; un quatrième tableau de I-1 mémoires est alors nécessaire. Les lettres interdites ne sont pas supprimées dans les facteurs au cours des méthodes précédentes.

Pour gagner du temps, on peut écrire les facteurs restants en queue de file ; on perd ainsi évidemment de la place (cette variante est programmée en annexe). Si les lettres interdites dans les facteurs sont supprimées, on peut réduire les facteurs (c'est d'autant plus intéressant que les facteurs sont nombreux vis-à-vis du nombre de lettres).

Remarque 3.

On peut donner un autre aspect à l'algorithme lexicographique indirect, en remplaçant la donnée d'un facteur $(b + B^+)$ par $1 + \mathrm{Card}\ B$ expressions de la forme B^b. Ainsi $(a + b + d)$ donnera naissance à ab^d, ad^b, bd^a. B^b signifiera : si toutes les lettres de B sont interdites, alors b est nécessaire. Les ensembles de la forme B (ensembles des lettres des facteurs moins une) seront appelés paquets. On peut énoncer :

$B \cup C \cup \ldots \cup D \subset E \cup F \cup \ldots \cup G$ entraîne : si e, f, ..., g sont interdits, alors b, c, ..., d sont interdits.

Cette propriété implique les trois cas particuliers suivants :

1. $B \subset E$ entraîne : si e est interdit, alors b est interdit

2. $B = E$ entraîne : e est interdit si et seulement si b est interdit ; un monôme premier du développement contenant e contient b et

réciproquement. La lettre e par exemple peut être supprimée dans le produit de sommes et b remplacée par eb. (b + e) n'est pas un facteur.

L'autre sorte de symétrie est : l'existence pour le développement d'un monôme premier $bM^{\wedge}$, où M ne contient pas la lettre e, entraîne l'existence du monôme premier $eM^{\wedge}$.

Un exemple de produit de facteurs ayant cette symétrie est : (a+b)(a+c)(b+c).

3. B = E ∪ F ∪ ... ∪ G entraîne : e, f, ..., g sont interdits si et seulement si b est interdit. Si l'une des lettres e, f, ..., g est nécessaire, alors b est nécessaire.

Pour un produit de facteurs à ensembles de lettres plutôt disjoints et de cardinaux élevés (ou, pour être plus rigoureux, disons que le nombre de facteurs est inférieur au nombre de lettres), si le test d'arrêt dans la construction d'une suite interdite est l'impossibilité d'adjoindre une nouvelle lettre ou la suppression de tous les facteurs, il est intéressant d'adjoindre les lettres interdites par paquets B, C, ... (le nombre d'adjonctions est au plus égal au nombre de facteurs) au lieu de lettre par lettre. Si le test d'arrêt dans la construction d'une suite interdite est la seule impossibilité d'adjoindre une nouvelle lettre, les lettres qui ne sont ni nécessaires, ni interdites quand il n'y a plus de facteurs, ne forment pas avec des lettres nécessaires un paquet ; l'exemple suivant le montre :

$$(a + x)(a + y)(x + y + z) = xy + ax + ay + az$$

Le produit de facteurs est équivalent à : a^{xy}, x^{a}, y^{a}, xy^{z}, xz^{y}, yz^{x}. L'interdiction de a entraîne la nécessité de x et y ; il ne reste alors plus de facteur ; z est à adjoindre comme lettre interdite alors que ni z ni az

ne sont des paquets.
En utilisant les paquets selon l'ordre décroissant du nombre de leurs lettres, on utilise les premier et deuxième cas particuliers cités au dessus.

Remarque 4.

Le premier monôme fourni par l'algorithme lexicographique indirect est premier, puisque toute lettre qui n'est pas nécessaire est interdite.

Remarque 5.

[GROSSI] donne une caractérisation géométrique des sommets caractéristiques de second espèce d'une fonction booléenne croissante.

Remarque 6.

Il est possible de généraliser l'algorithme lexicographique indirect au développement de produits de réunions de pavés (définis dans un produit de treillis distributifs) ; à condition de tenir compte des incompatibilités possibles entre pavés nécessaires et pavés interdits. Considérons par exemple le cas de produits de sommes de lettres complémentées ou non en algèbre de Boole ; si a est une lettre nécessaire alors a' est une lettre interdite et inversement ; mise à part cette implication qui accélère la détermination des lettres interdites, l'algorithme lexicographique indirect s'applique directement avec pour lettres les lettres complémentées et les lettres non complémentées.

2. Cas particulier d'un produit de sommes de deux lettres.

Désignons par m_a les lettres des facteurs contenant a qui sont différentes de a. L'algorithme précédent se simplifie de la façon suivante :

Algorithme

On forme, dans l'ordre lexicographique des lettres, les suites interdites ; dès qu'une lettre a est ajoutée à une suite interdite en formation, m_a est ajouté à la suite nécessaire en formation correspondante ; les lettres interdites et les lettres nécessaires sont distinctes.

Quand il n'existe plus de lettre pouvant être introduite dans la suite interdite, on forme la suite interdite qui la suit lexicographiquement si elle existe, ainsi que la suite nécessaire associée.

Le développement des facteurs de deux termes est, sans omission, ni répétition, l'ensemble des suites nécessaires telles que toute lettre n'y appartenant pas appartient à la suite interdite correspondante.

C'est l'algorithme utilisé dans [ARSAC] p. 103, [KUNTZMANN 3] p. 103 bis pour la détermination des ensembles intérieurement stables maximaux qui sont les suites interdites du développement de $\pi(x_i + x_k)$ (voir VI-4).

Un programme correspondant à cet algorithme est en annexe. Sa vitesse varie avec l'ordre des lettres adopté ; on constate qu'il est beaucoup plus rapide quand les lettres (a) sont ordonnées suivant l'ordre décroissant du nombre de lettres des ensembles (m_a) associés (des résultats précis sont en annexe). [CHATELIN] fournit pour les lettres un ordre légèrement différent et justifie théoriquement son efficacité.

c. Remarques.

1. Considérons l'algorithme lexicographique suivant, ressemblant au précédent mais tel que toute lettre non interdite lexicographiquement avant une lettre interdite est supposée nécessaire : on élimine du produit de facteurs la lettre nouvellement introduite dans la suite interdite en formation, puis on élimine les facteurs à une seule lettre soit x_ℓ ou contenant une

telle lettre x_ℓ, et on ajoute x_ℓ à la suite nécessaire en formation ; on ajoute aussi à la suite nécessaire en formation les lettres non introduites dans la suite interdite en formation et on élimine les facteurs les contenant, mais s'il n'y a pas d'élimination de facteur, on supprime les suites en formation et on passe aux suivantes. Quand il n'existe plus de facteurs, on forme la suite interdite qui suit lexicographiquement si elle existe ainsi que la suite nécessaire associée. Le développement des facteurs est l'ensemble réduit des suites nécessaires obtenues.

L'exemple précédent devient :

$$a^f b^d c^e + a^f b^d e^c + \overline{a^{bf} c^{ed}} + a^{fbc} + a_{bc}{}^d + a_b{}^c{}_{de}{}^f + a_b{}^c{}_{df}{}^e + \overline{a_b{}^{cd}} + \overline{a_c{}^{bd}} + abc$$

Cet algorithme est équivalent à l'algorithme direct de couverture minimale avec pour les lettres l'ordre inverse de l'ordre lexicographique, mais il n'est pas le même.

2. Efficacité comparée des algorithmes lexicographiques

L'algorithme différentié brut est trop combinatoire pour être rapide.

L'algorithme lexicographique indirect est sans doute plus rapide que l'algorithme lexicographique direct.

Quand les facteurs ont de nombreuses lettres et quand ils sont peu nombreux par rapport au nombre de lettres, l'algorithme lexicographique direct et la variante indirecte indiquée dans la remarque 3 permettent une élimination rapide de ces facteurs ; au contraire, quand ils ont peu de lettres et quand ils sont nombreux par rapport au nombre de lettres, leur élimination est rapide au cours des algorithmes lexicographiques indirects.

VI.3. Algorithmes progressifs de couverture minimale.

On développe certains facteurs, puis d'autres ... En particulier :

a. On peut développer les deux premiers facteurs, puis le résultat avec le troisième, puis le résultat avec le quatrième, ...[BENZAKEN 1] p. 116.

b. On peut développer les facteurs contenant a (soit différentiellement, soit progressivement), puis les facteurs contenant b (et ne contenant pas a),..., enfin développer les quantités ainsi calculées (le cas particulier des facteurs à deux termes est ainsi traité dans [BENZAKEN 1] p. 133 de manière approfondie).

c. Mais on peut aussi développer les facteurs contenant uniquement a et b, puis les facteurs contenant a, b et c (mais ne contenant pas uniquement a et b),..., enfin développer les quantités ainsi calculées ; c'est en fait le processus précédent après inversion de l'ordre des lettres et de l'ordre des facteurs pour le deuxième développement ; étudions en détail le cas du produit de facteurs de deux termes.

En adoptant les notations de [BENZAKEN 1] p. 133, une étape intermédiaire consiste à développer le produit

$$g(a+m)$$

où g est une somme de monômes réduits ne contenant pas la lettre a (les types I et II de monômes de g n'existent donc pas ; par contre le type III ou monômes de g multiples de m et le type IV ou monômes de g multiples ni de a ni de m existent). Donnons la méthode suivante de développement mettant à part non seulement les monômes de g multiples de m mais d'abord ceux qui sont diviseurs de m :

Algorithme.

1. On recherche dans g rangé en file un monôme diviseur de m (le premier de la file) ; s'il existe

* et s'il est égal à m, on l'échange avec le dernier monôme de la file ; la classe IV est alors l'ensemble des monômes de la file, le dernier excepté, on passe en 4 ;
* et s'il est différent de m, on adjoint m en queue de file ; la classe IV est l'ensemble des monômes de la file, ce monôme adjoint excepté ; on passe en 4 ;

2. On recherche les monômes multiples de m ; s'ilsexistent, ils sont extraits et adjointsen bout de file (ils constituent la classe III) ; les monômes restants constituent la classe IV ;

3. Les monômes de la classe IV sont multipliés par m et adjoints en queue de liste s'ils ne sont pas multiples des monômes de classe III ; l'ensemble de ces monômes adjointsest réduit ;

4. Les monômes de la classe IV sont multipliés par a .

Remarque 1.

L'algorithme est très rapide dans le cas où g a un monôme diviseur de m , puisqu'on évite alors les étapes 2 et surtout 3 qui est longue. On peut accroître l'utilité de l'étape 1 en ordonnant les lettres a suivant la cardinalité croissante des monômes m_a qui leur sont attachés (ou, de façon équivalente, suivant l'ordre croissant des degrés des sommets des sous-graphes correspondants) ; cependant, dans le cas le plus défavorable, cette étape demande autant de comparaisons que de monômes dans g pour aucun résultat ; elle peut donc être supprimée.

Remarque 2.

g étant le résultat du développement d'une certaine somme h de monômes (ne contenant ni a ni b) par b + p (où p ne contient ni a ni b)

$$g = h(b + p),$$

si m contient la lettre b, on peut ainsi simplifier l'étape 2 de l'algorithme : on recherche les monômes multiples de m parmi les seuls monômes obtenus dans l'itération précédente par multiplication par b.

Pour utiliser au maximum cette remarque, les lettres seront ordonnées de façon à former une couverture minimale de chemins élémentaires (et si possible un chemin hamiltonien) du graphe associé au produit de facteurs. Si m ne contient pas b, c'est l'étape 1 qu'il est possible de simplifier.

d. Remarque.

Les algorithmes progressifs se généralisent immédiatement aux expressions quelconques de sommes et de produits (produit de sommes de produits, somme de produit de sommes, ...)

e. Comparaison des algorithmes lexicographiques et des algorithmes progressifs.

2. D'un point de vue place en mémoire, les algorithmes progressifs ont l'intérêt de ne pas nécessiter le traitement simultané de tous les facteurs ; ils ont l'inconvénient de pouvoir faire apparaître de nombreux monômes intermédiaires. C'est l'inverse pour les algorithmes progressifs.

1. Contrairement aux algorithmes progressifs, les algorithmes lexicographiques ont l'intérêt de fournir en cours de fonctionnement des monômes premiers (c'est certain pour l'algorithme lexicographique indirect) ou presque premiers.

3. D'un point de vue rapidité, quand les facteurs ont peu de lettres et quand ils sont nombreux par rapport au nombre de lettres, l'algorithme lexicographique indirect est plus rapide que l'algorithme progressif. Dans le cas particulier de produits de sommes de deux lettres et avec la terminologie de V-3b (à chacune des sommes de deux lettres correspondent deux 1 de la matrice M dont les lignes et colonnes sont caractérisées par ces deux lettres), l'algorithme progressif est meilleur pour les matrices creuses (avec peu de 1), l'algorithme lexicographique indirect est meilleur pour les matrices pleines.

4. Pour rechercher les seules couvertures de cardinal minimal (c'est-à-dire les couvertures dont le nombre de termes est minimal), on délaisse évidemment toute suite de termes de cardinal supérieur (ou égal si une seule couverture de cardinal minimal est recherchée) au plus petit cardinal des couvertures déjà trouvées [BENZAKEN 1]. Il est alors intéressant de panacher les algorithmes lexicographique indirect et progressif, le premier servant à déterminer une valeur très proche du cardinal minimal, le deuxième déterminant les couvertures de cardinal minimal.

VI.4. Applications des algorithmes de couverture minimale.

Les algorithmes de couvertures minimale ont été l'objet de nombreuses études et ont de nombreuses applications ; en particulier, ils permettent de déterminer

- la fonction duale, la fonction complément et l'ensemble des monômes premiers d'une fonction booléenne ;
- les bases premières irredondantes d'une fonction booléenne, en développant les couvertures de monômes premiers de chaque point ou par la méthode des consensus [KUNTZMANN 2] p. 122 - 126 ;

- les ensembles extérieurement stables (ou sous-ensembles de sommets contenant tout sommet ou l'un de ses successeurs immédiats) minimaux d'un graphe orienté [MAGHOUT] : ils sont le produit du développement de

$$\prod_{x_i} (x_i + \Sigma \text{ sommets successeurs immédiats de } x_i) \qquad (1)$$

où $\{x_i\}$ est l'ensemble des sommets du graphe ;

- les ensembles fortement stables (ou sous-ensembles de sommets tels que deux sommets ne soient jamais adjacents, c'est-à-dire tels que deux sommets ne peuvent pas appartenir à une même arête [BERGE 2]) maximaux d'un hypergraphe ; on recherche leurs compléments par rapport à l'ensemble des sommets de l'hypergraphe en développant :

$$* \quad \prod_{E_j} \sum_{x_i \in E_j} \pi \text{ sommets de } E_i \text{ différents de } x_i \text{ s'il en existe, sinon 1}$$

où $\{E_j\}$ est l'ensemble des arêtes de l'hypergraphe et $\{x_i\}$ son ensemble de sommets ;

$$* \text{ ou } \prod (x_i + x_k)$$

pour tous les couples (x_i, x_k) de sommets de chaque arête de l'hypergraphe.

On recherche donc les ensembles fortement stables maximaux en développant

$$\bigcap (x'_i + x'_k)$$

pour tous les couples (x_i, x_k) de sommets de chaque arête de l'hypergraphe, si x'_i et x'_k désignent les sommets de l'hypergraphe excepté respectivement

x_i et x_k , $\cap$ le plus petit commun diviseur ou l'intersection ensembliste.

- les cliques (ou sous-ensembles de sommets deux à deux adjacents) maximales d'un hypergraphe, problème identique au précédent ; une approche géométrique de la recherche des cliques et ensembles fortement stables a été donnée en V-3b.

- Les ensembles stables (ou sous-ensembles de sommets ne contenant aucune arête non réduite à un point [BERGE 2]) maximaux d'un hypergraphe ; on recherche leurs compléments par rapport à l'ensemble des sommets ou sous-ensembles de sommets contenant au moins un sommet de chaque arête non réduite à un point, en développant :

$$\prod_{E_j \text{ de cardinal} > 1} \sum_{\text{sommets de } E_j}$$

- le nombre chromatique fort d'un hypergraphe ou le plus petit nombre de couleurs tel que toute arête ait tous ses sommets de couleur différente [BERGE 2] ; étant donné que c'est aussi le nombre minimal d'ensembles fortement stables couvrant l'ensemble des sommets de l'hypergraphe, il s'obtient à partir des ensembles fortement stables maximaux en appliquant une fois un algorithme de couvertures minimales et donc à partir de l'hypergraphe même en appliquant deux fois un tel algorithme ;

- le nombre chromatique d'un hypergraphe ou le plus petit nombre de couleurs tel qu'aucune arête non réduite à un point n'ait tous ses sommets de la même couleur [BERGE 2] ; c'est encore le nombre minimal d'ensembles stables couvrant l'ensemble des sommets ; il s'obtient donc à partir des ensembles stables par un algorithme de couvertures minimales ;

- la classe chromatique d'un hypergraphe ou le nombre chromatique de l'hypergraphe dont les sommets sont les arêtes du graphe initial et dont deux sommets sont adjacents si les arêtes dont ils sont les images sont adjacentes ; la classe chromatique forte d'un hypergraphe ou le nombre chromatique fort de l'hypergraphe associé ;

- les couplages (ou sous-ensembles d'arêtes tels que deux arêtes ne soient jamais adjacentes) maximaux d'un hypergraphe : on recherche leurs compléments par rapport à l'ensemble des arêtes en développant :

$$* \prod_{x_i} \sum_{E_j \supset x_i} \quad \prod \text{ arêtes contenant } x_i \text{ et différentes de } E_j \text{ s'il en existe, sinon 1}$$

$$* \text{ ou } \prod (E_i + E_k)$$

pour tous les couples (E_i, E_k) d'arêtes adjacentes de l'hypergraphe.

On recherche donc les couplages maximaux en développant

$$\bigcap (E'_j + E'_k)$$

pour tous les couples (E_j, E_k) d'arêtes adjacentes de l'hypergraphe ; E_j' et E_k' désignent les arêtes de l'hypergraphe excepté respectivement E_j et E_k.

- les supports (ou sous-ensembles de sommets tels que toute arête de l'hypergraphe a au moins un sommet dans le support) minimaux d'un hypergraphe, en développant

$$\prod_{E_j} \sum \text{ sommets de } E_j$$

- les <u>couvertures</u> ou <u>recouvrements</u> (ou sous-ensembles d'arêtes qui couvrent tous les sommets) minimaux d'un hypergraphe, en développant

$$\prod_{x_i} \sum \text{ arêtes contenant } x_i$$

- les ensembles intérieurement stables (appelés aussi parfois <u>sous-réseaux déconnectés</u>) maximaux d'un graphe [MAGHOUT], [BENZAKEN] puisque ce sont des ensembles stables et fortement stables selon la terminologie des hypergraphes ; les <u>noyaux</u> (ou ensembles stables intérieurement et extérieurement) maximaux puisque ce sont les ensembles intérieurement stables maximaux [BERGE 1] contenant au moins une lettre de chacun des facteurs de (1) ;

Citons aussi [LEMPEL, CEDERBAUM] où est appliqué un algorithme de couverture pour déterminer les sous-ensembles d'arcs et de sommets d'un graphe qui, supprimés, laissent le graphe résultant sans circuit.

REFERENCES BIBLIOGRAPHIQUES

CHAPITRE 1

BENZAKEN, C. Synthèse des réseaux de contact. Séminaire de logique, Grenoble, décembre 1965.

CURTIS, H. Allen. A new approach to the design of switching circuits. Van Nostrand, 1962, 635 p.

DESCHIZEAUX, P. Application à l'algèbre de Boole de la notion de dérivée. Séminaire de logique, Grenoble, février 1966.

KUNTZMANN, J. Algèbre de Boole. 2ème éd., Dunod, 1968, 361 p.

KUNTZMANN, J., LAPSCHER, F., PICHAT, E. Revue A Tijdschrift, Bruxelles, vol. IX, n° 3, juillet 1967, p. 119-120.

LAPSCHER, F. |1|. Propriétés relatives à la représentation des fonctions. Séminaire de logique, Grenoble, décembre 1965.

LAPSCHER, F. |2|. Application de la notion de fermeture à l'étude des fonctions booléennes. Thèse, Grenoble, septembre 1968, 273 p.

LEMPEL, A., CEDERBAUM, I. On the Minimal Synthesis of one Terminal-Pair Contact Networks. IEEE Trans., vol. CT-13, no-2, p. 149-53, juin 1966.

PICHAT, E. |1|. Décompositions des fonctions booléennes. Thèse, Grenoble, janvier 1966.

PICHAT, E. |2|. Décompositions simples de fonctions booléennes. R.I.R.O., n° 7, 1968, p. 61-70.

PICHAT, E. |3|. Des décompositions disjointes simples des fonctions booléennes. Séminaire de logique, Grenoble, mai 1967, 18 p.

PICHAT, E. |4|. Décompositions et écritures minimales des fonctions booléennes. Colloque "Les Techniques de Calcul et les Calculateurs", Bucarest, 22-26 septembre 1967, paru dans le Bulletin mathématique de la Société Scientifique Mathématique de la R.S. de Roumanie, tome 12 (60), n° 2, 1968, p. 143-151.

PICHAT, E., DESCHIZEAUX, P. Nouvel algorithme pour la décomposition simple des fonctions booléennes. Séminaire de logique, Grenoble, décembre 1967, 14 p.

SESHU, S., REED, M.B. Lineal Graphs and Electrical Networks. Addison-Wesley, 1961.

CHAPITRE 2

ASHENHURST, R. The decomposition of switching functions. Ann. Harvard Comput. Lab., 29, 1959, p. 74-116.

CURTIS, H. Allen. A new approach to the design of switching circuits. Van Nostrand, 1962, 635 p.

KARP, Richard M. Functional decomposition and switching circuit design. J. Soc. Indust. Appl. Math., 11, n° 2, juin 1963, p. 291-335.

KUNTZMANN, J. Algèbre de Boole. 2ème éd., Dunod, 1968, 361 p.

PICHAT, E. |1|. Décompositions des fonctions booléennes. Thèse, Grenoble, janvier 1966.

PICHAT, E. Décompositions d'une fonction de variables à nombres finis de valeurs. C.R. Acad. Sc. Paris, 267, série A, 1968, p. 761-763.

PICHAT, E. Décompositions d'une fonction de plusieurs variables. C.R. Acad. Sc. Paris, 268, série A, 1969, p. 1523-1526.

THELLIEZ, S. Sur la décomposition disjonctive complexe arborescente d'une fonction ternaire en vue de son application à la synthèse des structures combinatoires ternaires. C.R. Acad. Sc. Paris, 264 série A, 27 février 1967, p. 419-421.

CHAPITRE 3

BENZAKEN, C. Programme de calcul des monômes premiers d'une fonction booléenne. Institut de Mathématiques Appliquées, Grenoble, 1964.

BOURBAKI, N. Eléments de mathématique. Ensembles ordonnés. Fascicule XX, 2ème éd., Hermann, 1963, 148 p.

DUBREIL, P., DUBREIL - JACOTIN, M.L. Leçons d'algèbre moderne. Dunod, 1961, 393 p.

DUBREIL - JACOTIN, M.L., LESIEUR, L., CROISOT, R. Leçons sur la théorie des treillis, des structures algébriques ordonnées et des treillis géométriques. Gauthier-Villars, 1953, 385 p.

HARRAND, Y. Traitement dés files et des listes. Dunod, 1967, 120 p.

KUNTZMANN, J. Algèbre de Boole. 2ème éd., Dunod, 1968, 361 p.

TISON, P. Théorie des consensus. Thèse, Grenoble, 1965.

CHAPITRES 4, 5, 6 et 7

ARSAC, BERSTEL, FAULLE, GIRAULT, JACQUES, LENORMAND, PERROT, PETIT. Contrat Graphes. Université de Paris, 210 p.

BALINSKI, M.L. Integer programming : methods, uses, computation. Management Science, vol. 12, n° 3, 1965, p. 253-313.

BECKENBACH, E.F. Applied combinatorial mathematics. Wiley, 1964, 608 p.

BEDNAREK, A.R., TAULBEE, O.E. On maximal chains. Rev. roum. math. pures et appl., 1966, tome XI, n° 1, p. 23-25.

BENZAKEN, C. Algorithmes de dualisation d'une fonction booléenne. RFTI Chiffres, 9, n° 2, 1966, p. 119-128.

BENZAKEN, C. Structures algébriques des cheminements : pseudo-treillis, gerbiers de carré nul. p. 40-47 dans Network and switching theory, edited by G. BIORCI, Academic Press, 1968, 634 p.

BENZAKEN, C. [1]. Contribution des structures algébriques ordonnées à la théorie des réseaux. Thèse, Grenoble, 1968, 216 p.

BERGE, C. |1|. Théorie des graphes et ses applications. 2ème éd., Dunod, 1967, 270 p.

BERGE, C. |2|. Sur certains hypergraphes généralisant les graphes bipartites. I.R.I.A., 78-Rocquencourt, 1969, 20 p.

BOUCHET, A. Recherche des éléments hamiltoniens d'un réseau. Séminaire de logique, Grenoble, 1967.

CHATELIN, Ph. Application du langage Cassandre à la microprogrammation. Contrat DRME, Université de Grenoble, 1970, à paraître.

CHATELIN, P., PICHAT, E. Minimisation de la partie commande d'une mémoire microprogrammée. Grenoble, workshop on microprogramming, 9-10 juin 1970.

CHEIN, M. Etude des décompositions d'un réseau. Thèse, Grenoble, 1967, 99 p.

COURTIN, J. Langages analysables de gauche à droite : construction d'un analyseur pour langages LR(1). Thèse, Grenoble, 1968, 179 p.

FALKOFF, A.D., IVERSON, K.E., SUSSENGUTH, E.H. Formal description of system /360, IBM Sys. J., vol. 3, 1964, p. 200.

DUBREIL-JACOTIN, M.L., LESIEUR, L., CROISOT, R. Leçons sur la théorie des treillis, des structures algébriques ordonnées et des treillis géométriques. Gauthier-Villars, Paris, 1953, 385 p.

GILLI, L., MEO, A.R. Algorithmi per la minimizzazione di funzioni logiche. R.C. Riunione A.E.I., Alghero, 1966, III, n° 66, 26 p.

GRASSELLI, A., LUCCIO, F. Some coveringproblems in switching theory. p. 536-557 dans Network and switching theory, edited by G. BIORCI, Academic Press, 1968, 634 p.

GRASSELLI, A., MONTANARI, U. On the minimization of read - only memories in microprogrammed digital computers. Note interne B68/17, Instituto di elaborazione dell'informazione, Pisa, octobre 1968, 13 p.

GRIFFITHS, M. Analyse déterministe et compilateurs. Thèse, Grenoble, 1969, 81 p.

GROSSI, A. Thèse de 3ème cycle, Grenoble, à paraître.

HAMMER, P.L., RUDEANU, S. Boolean methods in operations research. Springer-Verlag, 1968, 329 p.

HARRISON, Michael A. Introduction to switching and automata theory. McGraw Hill, 1965, 499 p.

HU, T.C. Revised matrix algorithms for shortest paths. SIAM J. Appl. Math., vol. 15, n° 1, janvier 1967, p. 207-218.

KAUFMANN, A., MALGRANGE, Y. Recherche des chemins et circuits hamiltoniens d'un graphe. Revue française de recherche opérationnelle, 7ème année, n° 26, 1963, p. 61-73.

KAUFMANN, A. |1|. Methodes et modèles de la recherche opérationnelle. Tome 2, 2ème éd., Dunod, 1968, 544 p.

KAUFMANN, A. |2| [2]. Introduction à la combinatorique en vue des applications. Dunod, 1968, 609 p.

KUNTZMANN, J. |1|. Théorie des relations et des réseaux. Cours, Université de Grenoble, 1966-1969.

KUNTZMANN, J. |2|. Algèbre de Boole. 2ème éd., Dunod, 1968, 361 p.

LAWLER, E.L. Covering problems : duality relations and a new method of solution. SIAM J. Appl. Math., vol. 14, n° 5, septembre 1966, p. 1115-1132.

LEMPEL, A., CEDERBAUM, I. Minimum feedback arc and vertex sets of a directed graph. p. 447-457 dans Network and switching theory, edited by G. BIORCI, Academic Press, 1968, 634 p.

LUNC, A.G. Izv. Akad. Nauk SSSR, 16, 1962, p. 405-426.

MAGHOUT, K. Applications de l'algèbre de Boole à la théorie des graphes et aux programmes linéaires et quadratiques. Cahiers du Centre d'études de recherche opérationnelle, vol. 5, n° 1-2, 1963, p. 21-99.

MALGRANGE, Y. Recherche des sous-matrices premières d'une matrice à coefficients binaires. Application à certains problèmes de graphe. p. 231-242 dans Deuxième congrès de l'AFCALTI, octobre 1961, Gauthier-Villars, 1962, 524 p.

McCLUSKEY, E.J. Minimization of boolean functions. Bell System Tech. J., 35, 1956, p. 1417-1444.

McNAUGHTON, R., YAMADA, H. Regular expressions and state graphs for automata. IRE Trans. electronic computers, EC-9, 1960, p. 39-47.

MEO, A.R. On the synthesis of many-variable switching functions. p. 470-482 dans Network and switching theory, edited by G. BIORCI, Academic Press, 1968, 634 p.

MOREALE, E. Partitioned lists techniques in Quine's method implementation. p. 483-495 dans Network and switching theory, edited by G. BIORCI, Academic Press, 1968, 634 p.

ORE, O. Theory of graphs. American Mathematical Society, vol. 38, 1962.

PAIR, C., DERNIAME, J.C. Etude des problèmes de cheminement dans les graphes finis. Convention DGRST n° 66-002, 164 p.

PAULL, M.C., UNGER, S.H. Minimizing the number of states in incompletely specified sequential switching functions. IRE Trans. Electro. Computers, 8, 1959, p. 356.

PICHAT, E. |1|. Décompositions simples de fonctions booléennes. R.I.R.O., n° 7, 1968, p. 61-70.

PICHAT, E. |2|. Algorithms for finding the maximal elements of a finite universal algebra. Proc. IFIP Congress 1968, Edinburgh, booklet A, p. 96-101, ou Information processing 68, North-Holland, Amsterdam, 1969, p. 214-218.

PICHAT, E. Algorithmes pour rechercher les éléments maximaux de structures algébriques. Congrès international des mathématiciens, Nice, septembre 1970.

QUINE, W.V. A way to simplify truth functions. American Math. Monthly, vol. 62, 1955, p. 627-631.

ROBERT, P., FERLAND, J. Generalisation de l'algorithme de Warshall. R.I.R.O., 2ème année, n° 7, 1968, p. 71-85.

ROY, B. [1]. Cheminement et connexité dans les graphes. Applications aux problèmes d'ordonnancement. Metra, Paris, 2ème éd., 1965.

ROY, B. Algèbre moderne et théorie des graphes. Dunod, Paris 1969, 502 p.

RUDEANU, S. Axiomatization of certain problems of minimization. Studia logica, tom XX, 1967, p. 37-61.

SALOMAA, Arto. Theory of Automatica. Pergamon Press, 1969, 263 p.

TALANCE, X. de. Recherche du noyau d'un graphe sans circuit. Communication personnelle.

TISON, P. |1|. Recherche des bases premières. Automatisme, juin 1965, p. 229-234.

TISON, P. |2|. Fondement d'une algèbre des systèmes logiques. Automatisme, juillet-août 1967, p. 298-303.

TOMESCU, I. Sur les méthodes matricielles dans la théorie des réseaux. C.R. Acad. Sc. Paris, t. 263, série A, 1966, p. 826-829.

TOMESCU, I. |1|. Méthode pour la détermination de la fermeture transitive d'un graphe fini. R.I.R.O., 1ère année, n° 4, 1967, p. 33-37.

TOMESCU, I. |2|. Recent research in the theory of boolean matrices. Economic computation and economic cybernetics studies and research, Bucharest, 1969, n° 2, p. 51-64.

TOMESCU, I. |3|. Sur l'algorithme matriciel de B. Roy. R.I.R.O., 2ème année, n° 7, 1968, p. 87-91.

TREHEL, M. Un programme de simplication de polynômes de réseaux. R.I.R.O. 2ème année, n° 9, 1968, p. 91-96.

VEILLON, G. Application de l'algèbre desexpressionsrégulières à l'étude des cheminements dans les graphes finis. 2ème thèse, Grenoble, 1970, 11 p.

WARSHALL, S. A theorem on boolean matrices. J. of the A.C.M., Jan. 1962, p. 11-12.

YEN, Jin Y. Some algorithms for finding the shortest routes through the general networks. p. 213-221 dans Computing methods in optimizations problems-2, edited by ZADEH, NEUSTADT, BALAKRISHNAN (papers presented at a conference held at San Remo, Italy, september 9-13, 1968), Academic Press, 1969, 313 p.

INDEX DES TERMES ET NOTATIONS

La rédaction de ce travail suppose connu le livre d'algèbre de Boole de J. KUNTZMANN.

VU

Grenoble, le

Le Président de la Thèse

VU

Grenoble, le

Le Doyen de la Faculté des Sciences

VU, et permis d'imprimer

Le Recteur de l'Académie de GRENOBLE