Contribution à la théorie des catalogues

Michel Trehel

N° d'ordre

T H E S E

Présentée à la Faculté des Sciences

de l'Université de GRENOBLE

pour obtenir

le titre de Docteur de Troisième Cycle

"Mathématiques Appliquées"

par

Michel TREHEL

Licencié-es-Sciences

CONTRIBUTION A LA THEORIE DES CATALOGUES

Thèse soutenue le 26 novembre 1965 devant la Commission d'Examen

MM. KUNTZMANN Président

VAUQUOIS

GASTINEL Examinateurs

FACULTE DES SCIENCES

LISTE DES PROFESSEURS

DOYENS HONORAIRES M. FORTRAT P.
M. MORET L.

DOYEN M. WEIL L.

PROFESSEURS TITULAIRES

MM.	NEEL L.	MAGNETISME
	HEILMANN R.	CHIMIE ORGANIQUE
	KRAVTCHENKO J.	MECANIQUE RATIONNELLE
	CHABAUTY C.	MATHEMATIQUES PURES
	PARDE M.	POTAMOLOGIE
	BENOIT J.	RADIOELECTRICITE
	CHENE M.	CHIMIE PAPETIERE
	BESSON J.	ELECTROCHIMIE
	WEIL L.	THERMODYNAMIQUE
	FELICI N.	ELECTROSTATIQUE
	KUNTZMANN J.	MATHEMATIQUES APPLIQUEES
	BARBIER R.	GEOLOGIE APPLIQUEE
	SANTON L.	MECANIQUE DES FLUIDES
	OZENDA P.	BOTANIQUE
	FALLOT M.	PHYSIQUE INDUSTRIELLE
	GALVANI O.	MATHEMATIQUES
	MOUSSA A.	CHIMIE NUCLEAIRE ET RADIOACTIVITE
	TRAYNARD P.	CHIMIE GENERALE
	SOUTIF M.	PHYSIQUE GENERALE
	CRAYA A.	HYDRODYNAMIQUE
	REULOS R.	THEORIE DES CHAMPS
	AYANT Y.	PHYSIQUE APPROFONDIE
	GALISSOT F.	MATHEMATIQUES PURES
Mlle	LUTZ E.	MATHEMATIQUES GENERALES
MM.	BLAMBERT M.	MATHEMATIQUES
	BOUCHEZ R.	PHYSIQUE NUCLEAIRE
	LLIBOUTRY L.	GEOPHYSIQUE
	MICHEL R.	GEOLOGIE ET MINERALOGIE
	BONNIER E.	METALLURGIE
	DESSAUX G.	PHYSIOLOGIE ANIMALE
	PILLET E.	ELECTROTECHNIQUE
	DEBELMAS J.	GEOLOGIE GENERALE
	GERBER R.	MATHEMATIQUES PURES
	PAUTHENET R.	ELECTROTECHNIQUE
	VAUQUOIS B.	CALCUL ELECTRONIQUE
	SILBER R.	MECANIQUE DES FLUIDES
	MOUSSIEGT J.	ELECTRONIQUE
	BARBIER J.C.	PHYSIQUE EXPERIMENTALE

	BUYLE-BODIN M.	ELECTRONIQUE
	KOSZUL J.L.	MATHEMATIQUES
	DREYFUS B.	THERMODYNAMIQUE
	VAILLANT F.	ZOOLOGIE
	KLEIN J.	MATHEMATIQUES PURES
	SENGEL P.	ZOOLOGIE
	ARNAUD P.	CHIMIE
	BARJON R.	PHYSIQUE NUCLEAIRE
	BARNOUD F.	BIOSYNTHESE DE LA CELLULOSE

PROFESSEURS ASSOCIES

MM.	WAGNER H.	BOTANIQUE
	NAPP-ZINN K.	BOTANIQUE

PROFESSEURS SANS CHAIRE

Mme	KOFLER L.	BOTANIQUE
	DEPASSEL R.	MECANIQUE
	PERRET R.	SERVOMECANISME
Mme	BARBIER M.J.	ELECTROCHIMIE
	COHEN J.	PHYSIQUE
	GIDON P.	GEOLOGIE
Mme	SOUTIF J.	PHYSIQUE GENERALE
	GIRAUD P.	GEOLOGIE
	GASTINEL N.	MATHEMATIQUES APPLIQUEES
	LACAZE A.	THERMODYNAMIQUE
	GLENAT R.	CHIMIE ORGANIQUE
	BRISSONNEAU P.	PHYSIQUE GENERALE
	DUCROS P.	MINERALOGIE
	ANGLES D'AURIAC	MECANIQUE DES FLUIDES
	ROBERT A.	CHIMIE PAPETIERE
	COUMES A.	ELECTRONIQUE
	PEBAY-PEROULA	PHYSIQUE
	DEGRANGE C.	ZOOLOGIE
	GAGNAIRE D.	CHIMIE PAPETIERE
	RASSAT A.	CHIMIE
	PERRIAUX J.	GEOLOGIE
	BARRA J.	MATHEMATIQUES APPLIQUEES

PROFESSEURS HONORAIRES

MM.	FORTIER A.	MECANIQUE DES FLUIDES
	BRELOT M.	MATHEMATIQUES
	WOLFERS F.	PHYSIQUE
	DORIER A.	ZOOLOGIE

MAITRES DE CONFERENCES

MM.	BIAREZ J.	MECANIQUE DES FLUIDES
	DODU J.	MECANIQUE DES FLUIDES

	DOLIQUE J.M.	ELECTRONIQUE
	HACQUES G.	MATHEMATIQUES APPLIQUEES
	LANCIA R.	PHYSIQUE AUTOMATIQUE
	POULOUJADOFF M.	ELECTROTECHNIQUE
	KAHANE A.	PHYSIQUE
Mme	BONNIER J.	CHIMIE
Mme	KAHANE J.	PHYSIQUE
	DEPORTES C.	CHIMIE MINERALE
	DEPOMMIER P.	PHYSIQUE NUCLEAIRE
	CAUQUIS G.	CHIMIE GENERALE
	BONNET G.	PHYSIQUE
Mme	BOUCHE L.	MATHEMATIQUES
	COLOBERT L.	PHYSIOLOGIE ANIMALE
	PAYAN J.J.	MATHEMATIQUES
	CAUBET J.P.	MATHEMATIQUES
	LAURENT P.	MATHEMATIQUES APPLIQUEES
	BERTRANDIAS J.P.	MATHEMATIQUES APPLIQUEES
	BRIERE G.	PHYSIQUE
	LAJZEROWICZ J.	PHYSIQUE
	VALENTIN J.	PHYSIQUE
	DESRE P.	METALLURGIE
	BONNETAIN L.	CHIMIE MINERALE

MAITRE DE CONFERENCES ASSOCIE

MM.	RADELLI L.	GEOLOGIE

Je tiens à exprimer ma profonde reconnaissance à

Monsieur le Professeur Kuntzmann qui a bien voulu me faire l'honneur de présider le Jury .

Monsieur le Professeur Vauquois qui a dirigé mon travail et m'a ouvert d'intéressantes voies de recherche.

Monsieur le Professeur Gastinel qui a accepté d'être membre du Jury.

Monsieur Veillon, Ingénieur au C.N.R.S. qui, par ses conseils, m'a permis d'orienter certaines études.

Messieurs Lecarme, Quezel-Ambrunaz, Madame Stiers, Monsieur Veyrunes pour leurs conseils en programmation.

Mademoiselle Anselmetti, Mademoiselle Curinier, Monsieur Mounet, Madame Pilon qui ont participé à la réalisation matérielle de cette thèse.

TABLE DES MATIERES

CHAPITRE -II- LES TRANSFORMATIONS

CHAPITRE -III- ALGORITHMES DES PROGRAMMES DE TRANSFORMATIONS ET DE MODIFICATIONS

I N T R O D U C T I O N

Nous voudrions ici présenter l'historique de la théorie des catalogues et noter en quoi elle diverge des études récemment faites sur les graphes et les arborescences.

Vers octobre 1963 il parut intéressant à des laboratoires de traduction automatique de standardiser un format de textes afin de rendre plus facile et plus rapide l'échange des bandes magnétiques. MM. VAUQUOIS et VEILLON (au Centre d'Etudes pour la Traduction Automatique de Grenoble) et MM. HAYS, KAY et ZIEHE (au Laboratoire de linguistique de la Rand Corporation à Santa-Monica) coordonnèrent leurs efforts dans ce sens.

Les seuls catalogues projetés alors ne comprenaient qu'une classe par niveau . Leur structure fut définie en mars 1964 par M. VEILLON (7) et la possibilité de leur emploi généralisé. On étendait l'écriture des textes à celle de quelconques informations.

Lors de son voyage en France en septembre 1964, M. KAY proposa l'idée des catalogues à plusieurs classes par niveau (4). Dès lors les formats de bandes magnétiques allèrent s'améliorant jusqu'en avril 1965 où fut envisagée la généralisation aux catalogues de catalogues.

De nombreux centres de traduction automatique se sont intéressés depuis au format de catalogues et sans doute des échanges auront-ils lieu dans un proche avenir.

Les catalogues sont des graphes arborescents mais leur originalité par rapport aux graphes jusqu'alors étudiés, réside dans leur description au moyen d'une carte qui est à la fois une arborescence à nombre de noeuds très limité et une sorte de résumé du catalogue.

Dans sa thèse de doctorat de philosophie à l'Université de Pennsylvanie, soutenue en 1963, M. HOLT (3) étudie les transformations d'arborescences. Les transformations de catalogues seront notre objectif principal et nous verrons comment les décrire à partir des transformations des cartes.

Notre étude renferme deux parties : l'une théorique concerne la définition et les propriétés des catalogues ; l'autre technique traite de bandes magnétiques et de programmes.

On ne trouvera pas dans les pages qui suivent la description rigoureuse du format de bandes. Cet article a été sciemment omis pour être réservé aux programmeurs qui s'intéresseront à l'aspect technique de cette thèse.

-I- DEFINITION DES CATALOGUES

A- RAPPELS - DEFINITION

1) Graphes à relation d'ordre

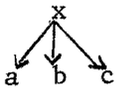
Certaines notations de la théorie des graphes seront utilisées Soient X un ensemble d'éléments, U un ensemble de relations binaires entre ces éléments.

x, a, b, c .... sont des éléments de X . Supposons que les seuls α tels que nous ayons la relation $x \rightarrow \alpha$ soient a, b, c. On a une application de X dans l'ensemble des parties de X et on note :

$$\Gamma(x) = \{ a, b, c \} \text{ ; Réf. (Berge) (1).}$$

Nous introduisons ici une modification par rapport à la théorie des graphes : L'écriture $\Gamma(x) = \{a,b,c\}$ traduit l'existence des relations binaires $x \rightarrow a$, $x \rightarrow b$, $x \rightarrow c$ et de plus, une relation d'ordre entre a,b,c. Nous exprimons cet ordre en plaçant b à droite de a et c à droite de b.

Un tel graphe sera appelé graphe à relation d'ordre :



Arborescence de racine x à relation d'ordre

C'est un graphe (X,U) à relation d'ordre tel que :

a) tout sommet différent de x est l'extrémité terminale d'un seul arc.

b) x n'est l'extrémité terminale d'aucun arc.

c) (X,U) est connexe.

Toutes les arborescences dont on parlera étant à relation d'ordre, on ne le précisera pas à chaque fois dans la suite de l'exposé.

Nous noterons $C \in \hat{\Gamma}(B)$ si C appartient à l'arborescence de racine B. Nous dirons aussi, dans ce cas que B est sous C ou que les deux éléments sont comparables.

2) Datum

C'est la représentation de faits ou d'idées dans une version formalisée capable d'être prise en charge ou manipulée par quelques procéd (Définition IFIP (7)). On ne confondra pas le datum avec l'information qui est le sens que l'on attribue au datum au moyen de conventions connues à partir de sa représentation.

Exemple de datum : Chaîne de caractères représentant un mot.

3) Classe de data

Ensemble fini de un ou plusieurs data. Pour chaque classe on définit un datum vide noté O et pour certaines classes un datum privé de sens noté $\square$. On appelle X l'ensemble des classes.

Notations

Les noms de classes seront des lettres majuscules. Les data ser des lettres minuscules. De plus a_1, a_2 .... seront des data de la classe A b_1, b_2 ... des data de la classe B.... etc.

Critère de rangement des data

A chaque classe est associée une fonction d'ordre φ : $\varphi(a_i) < \varphi(a_{i+1})$. Nous écrirons plus simplement $a_i < a_{i+1}$
On définit la fonction inverse $f(a_i)=i$.

Puissance d'une classe

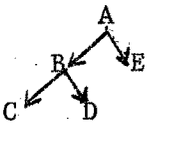
C'est le nombre de data que contient cette classe.

4) Carte d'un catalogue

On se servira de la définition d'une carte de catalogue pour définir et distinguer les divers catalogues.

Une carte de catalogue est une arborescence (X,U). En plus de leur nom, les classes auront un numéro. On numérote en partant de O de haut en bas puis de gauche à droite.

Ainsi dans la carte :



A	a le numéro	O
B	.....	1
C	.....	2
D	.....	3
E	.....	4

Niveau d'une classe

Le niveau N_x d'une classe de numéro x est :

1) 1 si x = 0

2) y + 1 si $x \in \Gamma^{1}(z)$ et $N_z = y$

- CATALOGUES SIMPLES

Ceux-ci ont été définis par M. VEILLON (7). Ils seront repris ici à partir de la notion de carte.

La carte d'un catalogue simple a une seule classe par niveau.

1) Article

Un article est une juxtaposition de data à raison d'un et d'un seul par classe. L'ordre des classes n'intervient pas dans un article.

2) Entrée

Une entrée est un article où l'ordre des data est l'ordre de leurs classes dans la carte. Une telle juxtaposition non commutative s'appelle une concaténation.

Sous-carte

Si X est un ensemble ordonné de noms de classes, une sous-carte est une partie de X avec la relation d'ordre induite.

Sous-entrée

C'est une entrée formée à partir d'une sous-carte donnée. En vue de comparer des entrées, nous allons définir le poids d'une classe, la valeur d'un datum, la valeur d'une entrée.

3) Comparaison et énumération d'entrées

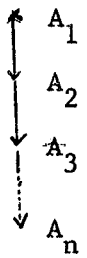
Poids d'une classe

Nous le définissons à partir d'une carte.

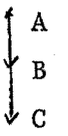
On note P(X) la puissance de la classe X et $\pi(X)$ le poids de la classe X.

On a $\pi(A_n) = 1$

$\pi(A_i) = \pi(A_{i+1}) \times P(A_{i+1})$



Exemple : Soit la carte



La classe A contient trois data $0, a_1, a_2$

La classe B contient deux data $0, b_1$

La classe C contient quatre data $0, c_1, c_2, c_3$

$\pi(C) = 1$ $\quad P(C) = 4$ $\quad \pi(B) = 4$

$P(B) = 2$ $\quad \pi(A) = 8$

Valeur d'un datum par rapport à une carte donnée

C'est le nombre $v(a_i) = f(a_i) \times \pi(A) = i \times \pi(A)$

La valeur d'un datum nul est 0. Pour tous autres data, la valeur nous permet de savoir la classe et le rang dans la classe, connaissant le poids des classes.

Valeur d'une entrée s

C'est la somme des valeurs des data composant cette entrée.

On la notera $v(s)$.

Enumération de deux entrées s_1 et s_2

C'est l'opération (s_1, s_2) telle que :

1) s_1 et s_2 sont composées des data de même classe.

2) $v(s_1) = v(s_2)$.

3) Si $v(s_1) = v(s_2)$ alors $(s_1, s_2) = s_1$.

cette opération est associative $((s_1, s_2), s_3) = (s_1, s_2, s_3)$

Catalogue simple

C'est une énumération d'entrées

Exemple : Avec les données de l'exemple précédent, on pourra avoir un catalogue tel que $a_1\ b_1\ c_1$, $a_1\ b_1\ c_3$, $a_2\ b_1\ c_3$.

Les data des classes A, B, C n'ont pas nécessairement tous été utilisé pour former le catalogue. Les classes en effet ne dépendent pas d'un catalogue particulier et on peut rencontrer les mêmes classes dans plusieurs catalogues sans rencontrer les mêmes data.

Distributivité

Soit l'énumération $a_i b_j$, $a_i b_k$; elle peut s'écrire $a_i(b_j,b_k)$. (b_j,b_k) est bien une énumération car $b_j < b_k$.

De même $a_i b_j$, $a_1 b_j = (a_i,a_1)b_j$.

Concaténation de deux énumérations

Soit $a_i b_j$, $a_i b_k$, $a_1 b_j$, $a_1 b_k$ une énumération

On peut encore l'écrire $a_i(b_j,b_k)$, $a_1(b_j,b_k)$

et encore $(a_i,a_1)(b_j,b_k)$

(a_i,a_1) et $(b_j b_k)$ sont des énumérations car $a_i < a_1$ et $b_j < b_k$.

4) Forme canonique et forme standard

Nous appellerons forme canonique, la forme sous laquelle on a défini le catalogue simple.

Cependant, on peut chercher à réduire cette forme en appliquant les règles de calcul que nous avons définies. En particulier, on peut écrire :
$C = \ldots a_i b_j \ldots e_k \ldots z_p , \ldots = \ldots , a_i\ C\ (a_i) , \ldots$
$C(a_i)$ étant les sous catalogues obtenus en supprimant a_i dans toutes les entrées le contenant. (Les sous catalogues sont relatifs à la carte obtenue en supprimant A de la carte de départ). La propriété de distributivité permet d'écrire une telle forme.

On peut appliquer le même procédé aux $C(b_j)$ dépendant de $C(a_i)$ et opérer ainsi jusqu'à ce que les dernières sous-entrées soient réduites à un datum.

La forme ainsi écrite est dite "forme standard" du catalogue. On la note

$$K = {}'a_i\ ({}'b_j \ldots ({}'_1\ z_1) \ldots)$$

Exemple : $a_1b_1c_1$, $a_1b_1c_2$, $a_1b_1c_3$, $a_1b_2c_4$

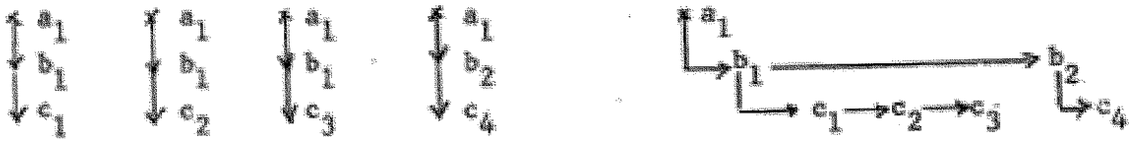
$= a_1\ (b_1c_1, b_1c_2, b_1c_3, b_2c_4)$

$= a_1\ (b_1\ (c_1,c_2,c_3)\ b_2c_4)$

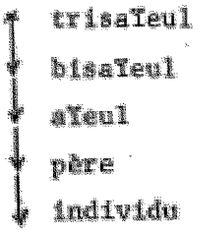
Nous allons dessiner le graphe de cet exemple et noter que les flèches verticales qui représentaient jusqu'alors des relations sur la carte pourront aussi représenter des relations entre data.

On a $\downarrow \begin{matrix} A \\ B \end{matrix}$ ou $\Gamma(A) = \{B\}$; $\downarrow \begin{matrix} a_1 \\ b_1 \end{matrix}$ ou $\gamma(a_1) = \{b_1\}$.

On utilisera des flèches horizontales pour exprimer la relation d'ordre entre des data de même classe gouvernés par un même datum.



Notons à ce propos que la carte d'un arbre généalogique est la suivante



Un arbre généalogique est exprimé sous forme standard, ce qui revient à dire qu'on ne répète pas le nom du trisaïeul bisaïeul .... pour chaque individu

5) Opérations sur les catalogues

Transformation

Cette opération consiste à changer la carte du catalogue en conservant les mêmes classes. Le catalogue transformé a les mêmes articles.

Modification

En gardant la même carte, cette opération consiste à ajouter des data, à en enlever ou en remplacer.

Nous verrons que ces opérations sont des cas particuliers de la transformation et modification des catalogues arborescents.

Nous étudions maintenant les opérations faisant intervenir plusieurs catalogues.

Concaténation de catalogues

Nous avons défini la concaténation d'énumérations. Les catalogues étant des énumérations, la concaténation peut s'y appliquer.

Soient $K_1 = e_1, e_2 \cdots e_n$ un catalogue associé à une carte C_1,

$K_2 = e'_1, e'_2 \cdots e'_n$ un catalogue associé à une carte C_2

C_1 et C_2 s'appliquent à des classes différentes.

Soit π_2 le produit des puissances de toutes les classes de C_2. On multiplie par π_2 les valeurs de toutes les entrées de K_1.

Les catalogues K_1 et K_2 sont alors des catalogues partiels sur les sous-cartes C_1 et C_2.

La concaténation de deux catalogues partiels K_1 et K_2 est la concaténation de deux énumérations de sous-entrées. On note $C_1\ C_2$ la carte résultant de l'énumération.

Union et intersection de deux catalogues

On peut unir deux catalogues qui ont la même carte. L'union est un catalogue dont les entrées appartiennent soit à l'un soit à l'autre. On définirait de même l'intersection.

Composition de deux catalogues

Soient C_1, C_2, C_3 trois cartes portant sur des classes différentes.

Soit K_1 un catalogue sur C_1, $K_1 = (e_1, \ldots e_n)$.

Soit e_i une entrée de K_1. A chaque valeur de i associons un catalogue K_2^i sur C_2 et K_3^i sur C_3.

On peut former n catalogues du type $e_i\ K_2^i$ sur la carte $C_1\ C_2$ et les énumérer en un catalogue noté $K_{12} = ,_i\ e_i\ K_2^i$.

De même on peut former n catalogues du type $e_i\ K_3^i$ et $K_{13} = ,_i\ e_i\ K_3^i$.

Enfin on peut former les catalogues $e_i\ K_2^i\ K_3^i$ et le catalogue $K_{123} = \sum_i e_i\ K_2^i\ K_3^i$.

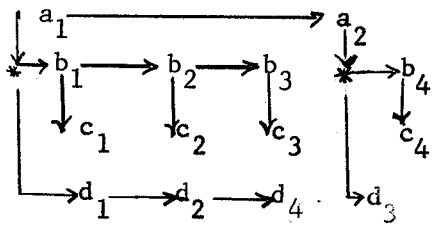
K_{123} est dit le composé de K_{12} et K_{13}.

Exemple : Soient les catalogues

$K_{12} = a_1b_1c_1,\ a_1b_2c_2,\ a_1b_3c_3,\ a_2b_4c_4$.

$K_{13} = d_1a_1,\ d_2a_1,\ d_3a_2,\ d_4a_1$.

Après transformation du second catalogue, on compose et on a :



✱ est le signe de composition.

La forme obtenue est appelée forme standard de mode I.

Condition pour que deux catalogues puissent être composés

La condition nécessaire et suffisante est que l'ensemble des sous-entrées par rapport à la carte constituée des classe $A_1, A_2 \ldots A_n$ soit le même dans les deux catalogues.

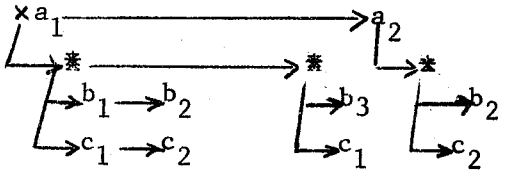
S'il n'y a pas de classe commune, la composition est la concaténation.

S'il y a des classes communes et que les sous-entrées ne sont pas les mêmes, on peut toujours imaginer des sous-entrées nulles pour rendre possible la composition.

Formes non standard de mode 2

Ce sont toutes les formes non standard différentes de celles de mode

Par exemple :



Nous verrons bientôt que ces formes non standard s'écrivent plus facilement avec des cartes qui ont plus d'une classe par niveau.

CATALOGUES ARBORESCENTS

1) Définition

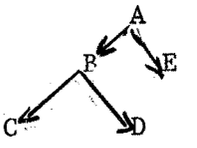
On appelle catalogue arborescent la donnée

a) de la carte.

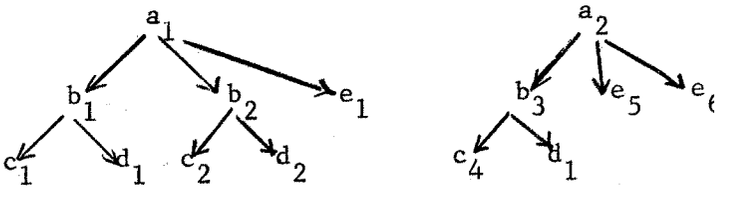
b) d'un ou de plusieurs data de classe 0.

c) pour toute classe A de la carte, de $\sigma(a_i) = \{(u_{a_i}), (v_{a_i}) \ldots (z_{a_i})\}$.
correspondant à $\Gamma(A) = \{U, V, \ldots Z\}$
$(u_{a_i}), (v_{a_i}) \ldots (z_{a_i})$ étant des énumération de data.

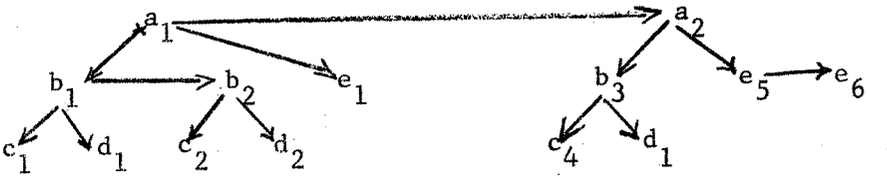
Exemple : Soit la carte :



un catalogue peut-être par exemple :



Nous l'écrirons



Les flèches horizontales représentent la relation d'ordre à l'intérieur d'une même classe.
Les flèches verticales, écrites ou omises (comme celles qui vont de a_1 à b_2 ou de a_2 à e_6) représentent les relations : Γ sur les classes , σ sur les data.

Dans l'exemple, a_1 sera appelé le gouvernant de b_1, b_2 et e_1, qui seront les dépendants de a_1; b_1 sera le premier dépendant.

Dans le cas particulier où la carte n'a qu'une classe par niveau, nous avons redéfini ci-dessus les formes standard, sinon, les formes non standard de mode 1.

2) Contenu d'un catalogue arborescent

A toute partie connexe du catalogue composée d'un datum de chaque classe, on fait correspondre un article formé des mêmes data.

Le contenu du catalogue est l'ensemble de ces articles.

3) Réalisation du catalogue à contenu minimum

On se donne une carte et un ensemble E_1 d'articles construits à partir de data des mêmes classes que la carte.

On veut réaliser un catalogue sur cette carte, dont le contenu est le plus petit ensemble d'articles contenant E_1.

On crée à partir de chaque article une partie connexe suivant la carte. Il y a alors un seul catalogue formé avec ces parties connexes.

Soit E_2 son contenu. On a $E_1 \subset E_2$. Démontrons que E_2 est minimum. Tout catalogue dont le contenu E_3 contient E_1 doit renfermer les parties connexes correspondant aux articles de E_1.

On a : $E_3 \supseteq E_2$

E_2 est donc minimum.

Note : Les catalogues dont on parle dans la suite de l'exposé, sont des catalogues arborescents.

D - CATALOGUES DE CATALOGUES

Les data n'ayant pas été limités en longueur, on rencontrera parfois, de très gros data dans certaines classes. Il pourra alors être intéressant d'envisager ces data comme de véritables catalogues.

Dans certaines classes, par conséquent, les data trouvés seront des catalogues. On peut aboutir ainsi à une récursivité d'ordre quelconque. Finalement l'ensemble du fichier se composera de data catalogues et de data simples.

⟨Fichier⟩ ::= ⟨début de fichier⟩ ⟨datum catalogue⟩ ⟨fin de fichier⟩
⟨datum catalogue⟩ ::= ⟨début de catalogue⟩ ⟨carte⟩ ⟨ensemble de data⟩ ⟨fin de catalogue⟩
⟨ensemble de data⟩ ::= ⟨datum⟩ | ⟨ensemble de data⟩ ⟨data⟩
⟨datum⟩ ::= ⟨datum simple⟩ | ⟨datum catalogue⟩

Cette écriture ne traduit pas le fait qu'on ne trouve des data catalogues que comme composant principal d'un fichier et dans certaines classes.

E - NOTES SUR LE FORMAT DE BANDES

1) Ordre des data sur une bande magnétique

Les data sont rangés de haut en bas puis de gauche à droite de man
à être toujours rencontrés avant leurs dépendants.

Ainsi dans l'exemple, on a : $a_1\ b_1\ c_1\ d_1\ b_2\ c_2\ d_2\ e_1\ a_2\ \ldots$

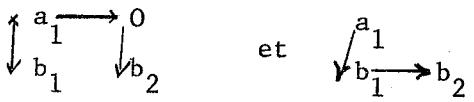
2) Data nuls

Nous avons vu dans la définition que a_i possédait au moins un dépendant de chaque classe U, V, ... Z. Ces dépendants pourront être des data nuls.

L'existence d'un datum nul signifie que nous n'avons pas d'informat à placer à un endroit donné au moment de la création du catalogue mais que nou pourrons plus tard le remplacer par un datum non nul.

Nous pouvons alors avoir un datum non nul gouverné par un datum nul Comme en général nous ne mettrons pas un datum nul sur bande magnétique, il nous faudra repérer ses dépendants pour éviter toute ambiguïté.

Représentons par 0 un datum nul, nous voyons que les graphes



sont différents.

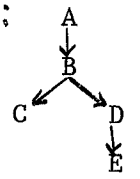
Les conventions d'écriture et les flèches nous permettent de repérer gouvernants et dépendants et de savoir que a_1 et 0 sont des data de la classe A, b_1 et b_2 de la classe B.

Sur bande magnétique, nous remplacerons ces informations en faisant accompagner chaque datum de son numéro de classe et de son numéro de niveau précédent que nous allons définir.

3) Numéro de niveau précédent (N.P.)

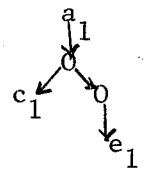
1) le N.P. d'un datum de niveau 1 est 0.
2) Si un datum est le premier dépendant d'un datum nul, son N.P. est le N.P. de ce datum nul.
3) Dans les autres cas, le N.P. d'un datum de niveau i est i-1.

Exemple : Soit la carte :



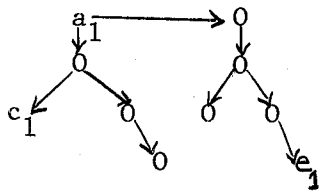
Niveau 1
Niveau 2
Niveau 3
Niveau 4

Nous donnons pour chaque datum son numéro de classe et de niveau précédent,



s'écrit $a_1(0,0)\ c_1(2,1)\ e_1(4,1)$.

Si on admet plusieurs data nuls par classe



s'écrit $a_1(0,0)\ c_1(2,1)\ e_1(4,0)$.

4) Interprétation du niveau précédent pour reconstituer le catalogue

A la rencontre d'un datum x, il nous faut pouvoir recréer tous les data nuls que contient le catalogue depuis le dernier datum non nul, y, rencontré.

x est de niveau i et de niveau précédent j

z, s'il existe, est le dernier datum de niveau j+1 rencontré, ou déjà créé s'il était nul.

Soit k le niveau de rattachement de x et y c'est-à-dire le niveau d'intersection des branches montantes depuis X et Y sur la carte.

Deux cas se présentent :

1) si $k > j$ on doit terminer l'arborescence de racine z, par des data nuls afin que toutes les classes de l'arborescence de racine Z soient représentées sous z par au moins un datum dans le catalogue. Puis il faut créer un datum nul de niveau $j + 1$ sauf si $i = j+1$, et ses descendants jusqu'à la classe X.

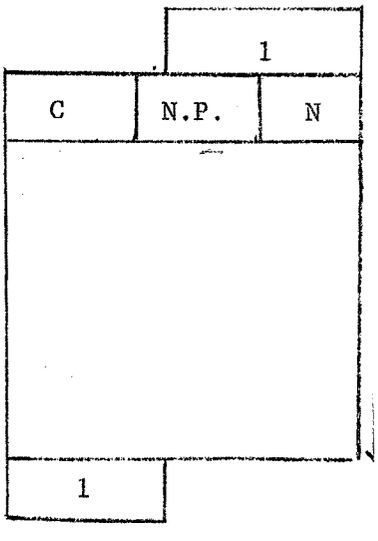
2) Si $k = j$ on doit créer des data nuls des classes $X + 1$ à $Y - 1$.

5) Arrangement des data

Nous avons vu que chaque catalogue est précédé de sa carte. Si c'est un catalogue de n classes, on trouvera n mémoires contenant le nom de classe, une information sur les data de la classe, nous permettant de savoir s'ils sont ou non des data catalogues, puis le numéro de niveau de la classe.

Les data simples sont précédés de préfixes indiquant leur longueur l, leur numéro de classe C, de niveau N et de niveau précédent NP, ils sont suivis suffixe indiquant leur longueur, en cas de nécessité de lecture arrière.

D'autre part pour conserver au système une grande souplesse, les data peuvent chevaucher su plusieurs enregistrements physiques ou être à plusieurs sur un même enregistrement.



En tête de chaque enregistrement physique se trouveront deux mots IØBS et FØRTRAN permettant d'utiliser les programmes de catalogues à divers niveaux de programmation. A Grenoble, nous traitons directement les programmes au niveau IOOP. Des procédures dépendantes du format lisent ou écrivent des data tandis que d'autres programmes agissent sur ces data pour transformer, modifier ou opérer diversement sur les catalogues.

-II- LES TRANSFORMATIONS

- DEFINITION D'UNE TRANSFORMATION T_{ij}

C'est une application de l'ensemble des catalogues de carte C_i dans l'ensemble des catalogues de carte C_j de mêmes classes que C_i, telle que, si K_i est un catalogue de carte C_i, le transformé $K_j = T_{ij}(K_i)$ est le catalogue à contenu minimum formé avec le contenu de K_i et la carte C_j.

On notera T_{ij} la transformation qui fait passer d'un catalogue de carte C_i à un catalogue de carte C_j et on dira par abus de langage que T_{ij} fait passer de C_i à C_j.

T_{ij} conserve le contenu si pour tout K_i construit sur C_i tel que $K_j = T_{ij}(K_i)$, le contenu de K_j est le même que celui de K_i.

Propriété I

Supposons que X soit une classe d'une carte C_1, autre que sa racine, et que C_2 soit l'arborescence de racine X. La structure obtenue en enlevant C_2 de C_1 est une arborescence que nous notons $C_1 - C_2$.

Soient E le gouvernant de X dans C_1, une carte C_3 de mêmes classes que C_1, et une carte C_4 de mêmes classes que C_2.

De plus $C_1 - C_2 = C_3 - C_4$.

Soit un catalogue K_1 construit sur C_1. Sous chaque datum de classe E, on a un catalogue de carte C_2 que nous transformons par T_{24}. On obtient le catalogue K_3 construit sur C_3.

Soit $K_2 = T_{13}(K_1)$.

Nous montrons que $K_2 = K_3$.

Il nous suffit de montrer que les catalogues de carte C_4 transformés des catalogues de carte C_2 se retrouvent dans K_2 sous les data de classe E.

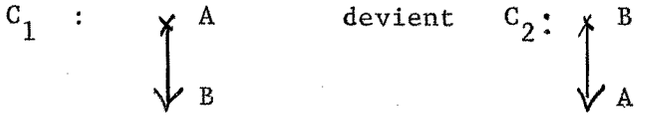
Ceci est évident en se reportant à la façon dont on construit le catalogue à contenu minimum :

Les parties connexes utilisées pour trouver K_3 appartiennent aux parties connexes utilisées pour trouver K_2.

Cette propriété nous permettra de ne faire des transformations que sur certaines parties des catalogues.

B - TRANSFORMATIONS ELEMENTAIRES

1) TE1



Le contenu est conservé

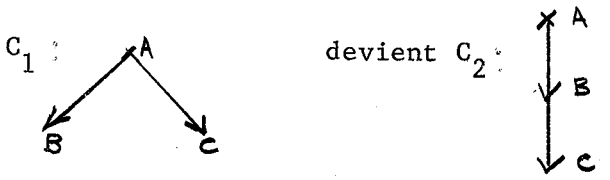
2) TE2

Il est clair qu'on peut définir simplement TE2 en écrivant :

Si $\Gamma(a_i) = \{(b_i), (c_i)\}$ par rapport à C_1,

alors $\Gamma(a_i) = \{(c_i), (b_i)\}$ par rapport à C_2.

3) TE3



Sous un même a, tous les c sont attachés à chaque b.

4) TE4

Tous les c appartenant à $\hat{\sqcap}$(a) vont appartenir à $\sqcap$(a).

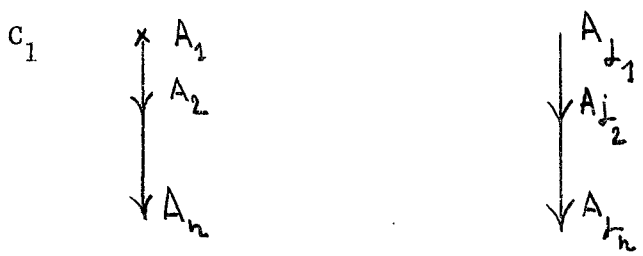
Remarque :

Les transformations TE1, TE2, TE3 conservent le contenu. Il n'en est pas de même pour TE4.

C - TRANSFORMATIONS ELEMENTAIRES GENERALISEES

Pour éviter un trop grand nombre de notations, nous les appellerons aussi, par abus de langage, TE1, TE2, TE3, TE4.

1) TE1



A_{j_1}, ... A_{j_n} est une permutation de A_1, A_2 ... A_n. Il n'y a aucune branche divergente de A_1 à A_n, ni de A_{j_1} à A_{j_n}.

De plus, $\hat{\Gamma}(A_n)$ est une structure quelconque qu'on retrouve dans C_2, A_n devenant A_{j_n}.

Si on considère le contenu du catalogue réduit à la carte qui va de A_I à A_n, on définira cette transformation en disant que ce contenu est conservé et que les structures attachées à ces articles avant la transformation leur restent attachées après.

2) TE2

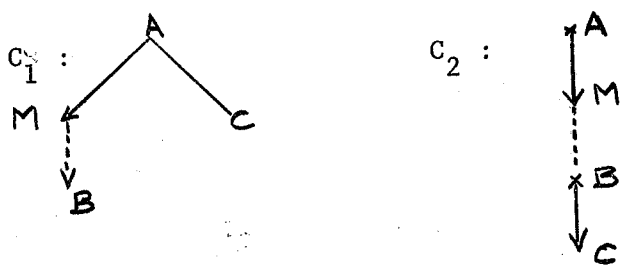
$\hat{\Gamma}(B)$ et $\hat{\Gamma}(C)$ sont des structures quelconques inchangées dans la transformation.

Si on avait $\Gamma(a_i) = \left\{ (b_i) , (c_i) \right\}$ avant la transformation, on aura

$\Gamma(a_i) = \left\{ (c_i) , (b_i) \right\}$ après.

Les structures qui étaient attachées aux b_i et c_i avant la transformation y seront attachées après.

3) TE3

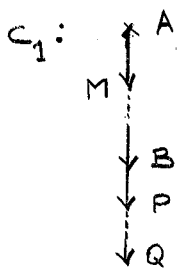


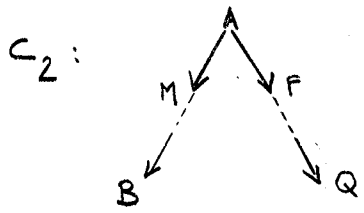
Dans C_1, M et C appartiennent à $\Gamma(A)$ et M est à gauche de C. B $\in \hat{\Gamma}(M)$.

L'arborescence de racine C devient un dépendant de B. Dans C_2, C suivra une classe D de $\hat{\Gamma}(B)$. On dira qu'on attache C à B derrière D.

Sous un même a, toutes les arborescences dont la racine est un datum de classe C, sont attachées à chaque b.

4) TE4



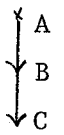


Dans C_1 on a $M \in \Gamma(A)$, $B \in \Gamma(M)$, $P \in \Gamma(B)$ et il n'y a pas de branches divergentes de P à Q. Dans C_2, M et P appartiennent à $\Gamma(A)$ et P est à droite de M.

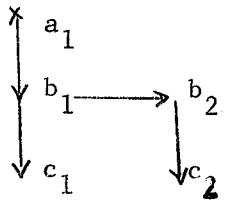
Sous un même a, l'ensemble des articles formés avec la branche qui va de P à Q est un catalogue qu'on rattache à a.

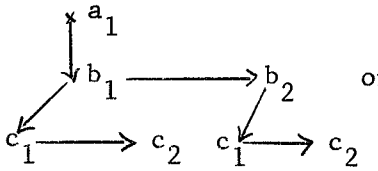
Par TE4, plusieurs catalogues différents peuvent avoir le même transformé.

Exemple

Si 

devient 



et


ont le même transformé

Nouvelle généralisation

Par la propriété 1, on dira que T_{13} est une transformation élémentaire s'il existe C_2 et C_4 arborescences des cartes C_1 et C_3 telles que :

1) $C_1 - C_2 = C_3 - C_4$.

2) Le passage de C_2 à C_4 est soit TE1, soit TE2, soit TE3, soit TE4

Produit de transformations

Soient T_{12} faisant passer des cartes C_1 à C_2 et T_{23} faisant passer des cartes C_2 à C_3. Si on a des catalogues K_1, K_2, K_3 tels que $K_2 = T_{12}\ (K_1)$ et $K_3 = T_{23}\ (K_2)$, on notera :

$K_3 = T_{23}.T_{12}\ (K_1)$

$T_{23} . T_{12}$ est un produit de transformations mais pas nécessairement une transformation.

Propriété 2

Si on passe d'une carte C_1 à une carte C_2 par un produit quelconque de transformations prises parmi TE1, TE2, TE3, le résultat obtenu est celui de la transformation T_{12}.

Soit un catalogue K_1 construit sur C_1, et E_1 son contenu. Par un produit de transformations prises parmi TE1, TE2, TE3 on arrive au catalogue K'_2 construit sur C_2 dont le contenu est encore E_1. E_1 est le contenu minimum d'un catalogue construit avec E1 et la carte C_2. C'est donc le contenu de $K_2 = T_{12}\ (K_1)$. On a $K_2 = K'_2$.

D - TRANSFORMATIONS REVERSIBLES . PROPRIETES

Partant d'une transformation T_{12} de C_1 à C_2, on va s'intéresser, dans ce chapitre, à la transformation T_{21} qui fait passer de C_2 à C_1.

1) Transformation réversible

T_{12} sera une transformation réversible si et seulement si, pour tout catalogue K construit sur C_1, on a :

$$T_{21} \cdot T_{12}(K) = K.$$

Transformation identique et inverse

Pour toute carte C_i et tout catalogue K construit sur C_i on a $T_{ii}(K) = K$. En effet K est le catalogue à contenu minimum construit avec la carte C_i et le contenu de K.

T_{ii} est donc un élément de l'ensemble de transformations identiques. On le note I.

Si T_{12} est réversible, on notera $T_{21} \cdot T_{12} = I$; T_{21} sera dit l'inverse à gauche de T_{12}. T_{12} sera l'inverse à droite de T_{21}.

Si $T' \cdot T = T \cdot T'' = I$, alors $T' = T''$; T' est appelé l'inverse de T.

Exemples

Les transformations TE1, TE2, TE3 sont réversibles.

L'inverse d'une transformation TE1 est une transformation TE1.

L'inverse d'une transformation TE2 est une transformation TE2.

L'inverse à gauche d'une transformation TE3 est une transformation TE4.

2) Produit de transformations réversibles

C'est une transformation réversible.

En effet, soient T_{12} et T_{23} réversibles : par définition, il existe T_{21} et T_{32}, telles que :

$$T_{21} \cdot T_{12} = I \quad \text{et} \quad T_{32} \cdot T_{23} = I \cdot$$

Pour tout catalogue K construit sur C_1, on a

$T_{32} \cdot T_{23} \cdot T_{12}$ (K) = T_{12} (K) ,

$T_{21} \cdot T_{32} \cdot T_{23} \cdot T_{12}$ (K) = $T_{21} \cdot T_{12}$ (K) = K .

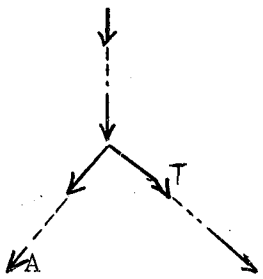
3) Le processus P

Ce processus, qui est un produit de transformations réversibles, remonte une classe quelconque à la racine d'une arborescence dont elle fait partie.

Nous décrivons le processus P sur un exemple avant d'en exposer l'algorithme.

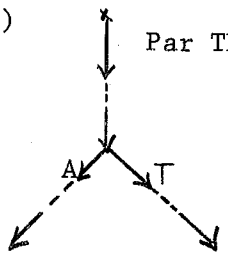
a) Exemple

Soit (1)

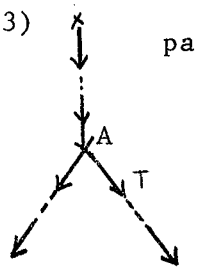


racine On désire remonter A à la de l'arborescence. On aura les phases suivantes :

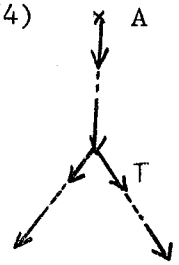
(2) Par TE1 puis



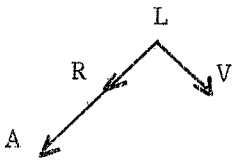
(3) par TE3 puis



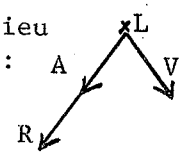
(4) A par TE1

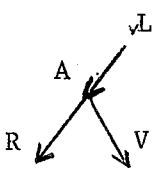


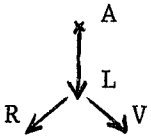
On note qu'on n'est pas passé directement de (1) à (3) ce qui aurait fait par exemple :



au lieu de :







Si on dit que le processus P est une transformation T_{12}, $V \in \overset{\wedge}{\Gamma}(R)$ dans C_2 entraîne $V \in \overset{\wedge}{\Gamma}(R)$ dans C_1.

Cette propriété sera appelée la propriété Q : elle est vraie pour tout couple de classes ne contenant pas la classe qu'on a remontée.

b) Description algorithmique

L'algorithme va nous permettre de trouver quelle est la prochaine opération à effectuer, à partir d'une situation quelconque pour réaliser le processus P.

Soit U le premier noeud à plusieurs dépendants, rencontré en remontant l'arborescence à partir de A.

2 cas se présentent :

1°- U n'existe pas (passage de (3) à (4) dans l'exemple).

Une transformation TE1 remonte A à la première place.

2°- U existe

- $A \notin \Gamma(U)$ Une TE1 remonte A parmi les éléments de $\Gamma(U)$ (passage de (1) à (2))

- $A \in \Gamma(U)$

- Si A suit immédiatement U dans la carte (Cf : numérotation des classes adoptée), soit $T \in \Gamma(u)$ et $T \neq A$. Une TE3 attache à A l'arborescence de racine T (passage de (2) à (3)).

- sinon, une TE2 met l'arborescence de racine A comme premier dépendant de U.

E - THEOREMES SUR LES TRANSFORMATIONS REVERSIBLES

Théorème 1

Pour qu'une transformation T_{op} de C_o à C_p, soit réversible, il faut et il suffit que les classes comparables dans C_o le soient aussi dans C_p.

Démonstration

1) Condition nécessaire de réversibilité

On fait ici une démonstration par l'absurde.

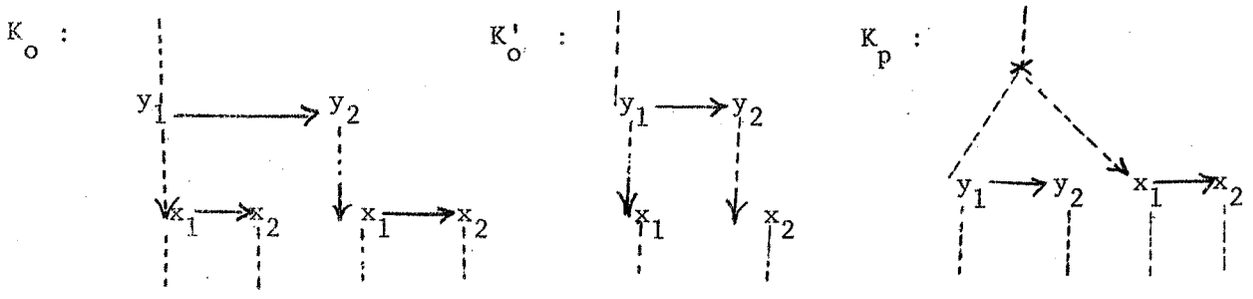
Soient deux classes X et Y comparables dans C_o, non dans C_p.

Trouvons deux catalogues K_o et K'_o construits sur C_o et ayant même transformé K_p. La transformation n'étant pas biunivoque ne pourra être réversible.

Ko et K'o, n'ont qu'un datum de chaque classe, excepté des classes X et Y où il y a deux data.

Dans K_o x_1 et x_2 appartiennent à $\hat{\Gamma}(y_1)$ et $\hat{\Gamma}(y_2)$.
Dans K'_o, $x_1 \in \hat{\Gamma}(y_1)$, $x_2 \in \hat{\Gamma}(y_2)$.

K_o et K'_o auront le même transformé K_p qui contiendra 4 parties connexes d'un datum de chaque classe.



2) Condition suffisante de réversibilité

Notre hypothèse est la suivante : les classes comparables dans C_o le sont aussi dans C_p.

Pour démontrer que T_{op} est réversible, nous passerons de C_o à C_p par une succession de transformations TE1, TE2, TE3. Supposons que C_o et C_p aient p classes.

Nous raisonnons par récurrence

a) Soit A la première classe de C_p

Nous faisons passer A à la racine de C_o par le processus P.

Nous notons que:

1°- On n'a fait que des transformations TE1, TE2, TE3.

2°- On a la propriété Q, sauf pour A

3°- Aucune classe ne doit sortir de l'arborescence de racine A

Cette dernière propriété est évidente car l'arborescence de racine A est la carte toute entière.

On appellera C_1 la carte obtenue.

b) Supposons qu'on ait pu attacher les n premières classes de C_p à leurs gouvernants.
On est arrivé à la carte C_n

Hypothèses de récurrence

1°- On n'a utilisé que TE1, TE2, TE3

2°- On a la propriété Q pour toutes classes, à l'exclusion des n premières, c'est-à-dire :
Si $I \in \hat{\Gamma}(J)$ dans C_n, c'est que $I \in \hat{\Gamma}(J)$ dans C_o

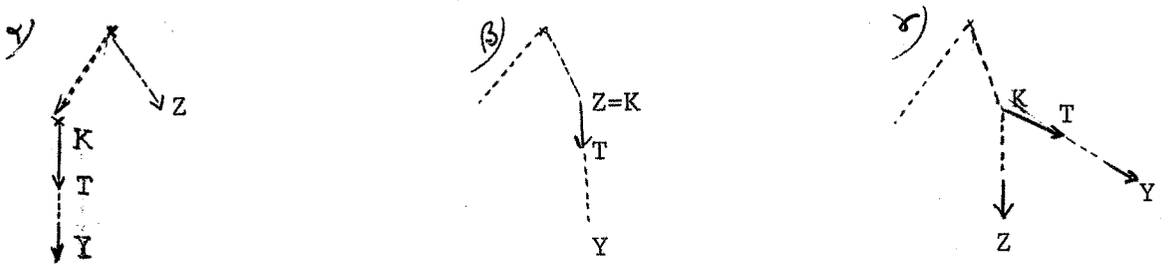
3°- Aucune classe ne doit sortir d'une arborescence dans laquelle elle se trouve et dont la racine est une des n premières classes

Cette propriété sera appelée R.

Rangeons la n + 1ème classe de C_p, Y, à partir de C_n pour obtenir C_{n+1}.

- Soit Z le gouvernant de Y dans C_p. Z est une des n premières classes de C_n.
- Soit K la classe de plus fort numéro, et appartenant aux n premières de C_n, telle que $Y \in \hat{\Gamma}(K)$.
  Soit T tel que $T \in \Gamma(K)$ et $Y \in \hat{\Gamma}(T)$.

On a dans Cn, une des trois situations suivantes :



α) Le cas α) est impossible car $Y \in \hat{\Gamma}(K)$ et par la propriété R, ne peut sortir de $\hat{\Gamma}(K)$

β) Le processus P remonte Y devant T, et éventuellement, une TE2 change l'ordre des dépendants de Z. Les propriétés Q et R ont le même sens pour C_n et C_{n+1} donc se conservent.

γ) Le processus P remonte Y devant T et une TE3 attache à Z la nouvelle arborescence de racine Y.

Q se conserve par la définition du processus P.

Montrons que R se conserve :

- Si $T \notin \hat{\Gamma}(Y)$ dans C_p,
  $Y \in \hat{\Gamma}(T)$ dans C_n donc dans C_o (Propriété Q)
  On ne peut avoir $Y \in \hat{\Gamma}(T)$ dans C_p.
  Y et T seraient comparables dans C_o, non dans C_p
  Ceci est impossible.
  Donc $T \in \hat{\Gamma}(Y)$ dans C_p

- Soit X une classe appartenant à $\hat{\Gamma}(T)$ dans C_n.
  $X \in \hat{\Gamma}(T)$ dans C_o (Q). Dans C_p on aura donc
  $X \in \hat{\Gamma}(T)$ ou $T \in \hat{\Gamma}(X)$. Donc $X \in \hat{\Gamma}(Y)$ dans C_p
  Aucune classe ne sortira de l'arborescence de racine Y.

  La propriété R se conserve.

c) Finalement nous sommes passés de C_n à C_{n+1} par un produit de transformations réversibles et en conservant Q et R.

On passe donc de C_o à C_p par un produit de transformations réversibles, donc T est réversible.

C.Q.F.D.

Corollaires

1) Toute transformation réversible est décomposable en un produit de transformations TE1, TE2, TE3.

2) Le produit est une opération interne et associative dans l'ensemble des transformations réversibles.

Théorème 2

Pour qu'une transformation T_{12} soit réversible, il faut et il suffit qu'elle conserve le contenu de tout catalogue construit sur C_1.

Démonstration

1°- Il faut qu'elle conserve le contenu.

En effet, la transformation T_{12} se décompose en un produit de transformations élémentaires qui conservent le contenu.

2°- Il suffit qu'elle conserve le contenu.

On sait que si elle n'est pas réversible, elle ne conserve pas le contenu.

Note : L'ensemble des transformations T_{op} telles que les classes comparables dans C_o et C_p soient les mêmes, forme un groupe pour le produit.

F - TRANSFORMATIONS IRREVERSIBLES

Nous nous intéressons dans ce chapitre à la décomposition d'une transformation irréversible en transformations élémentaires, comme nous l'avon fait pour les réversibles.

Propriété 3

Si un catalogue K_o de carte C_o et de contenu E_o se transforme par T_{op} en K_p de carte C_p et de contenu E_p, n'importe quel catalogue, de carte C_i de mêmes classes que C_o et C_p, et de contenu E_i tel que $E_o \subseteq E_i \subseteq E_p$, se transforme en Kp par T_{ip}.

Supposons en effet qu'un tel catalogue se transforme par T_{ip} en K'_p d'ensemble des articles E'_p.

$E_o \subseteq E_i$ entraîne $E_p \subseteq E'_p$.
$E_i \subseteq E_p$ entraîne $E'_p \subseteq E_p$.
Finalement $E_p = E'_p$.

Propriété 4

Si on peut passer de C_o à C_p par un produit de transformations élémentaires sans jamais détruire la comparabilité de deux classes comparables dans C_p, le résultat obtenu est celui de T_{op}.

Démonstration

Soient E_o, E_1 ... E'_p les contenus des catalogues

successifs obtenus en réalisant des transformations élémentaires. T_{op} transforme un catalogue de carte C_o en un catalogue de carte C_p, de contenu E_p.

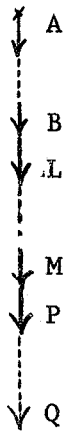
- Il suffit de montrer que $E_{p-1} \subset E_p$ d'après la propriété 3.
- Sachant que $E_i \subset E_p$, il faut montrer $E_{i+1} \subset E_p$.
- Nous faisons la démonstration pour E_o et E_1.

Si le passage du catalogue de contenu E_o à celui de contenu E_1 est réalisé par une transformation TE1, TE2 ou TE3, la propriété est évidente.

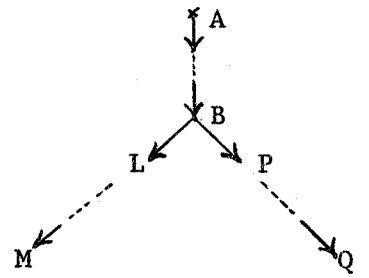
Sinon, nous utilisons TE4.

Rappelons quelles sont les cartes de départ et d'arrivée de TE4

C_o :

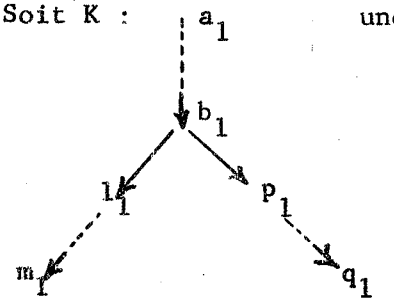


C_1 :



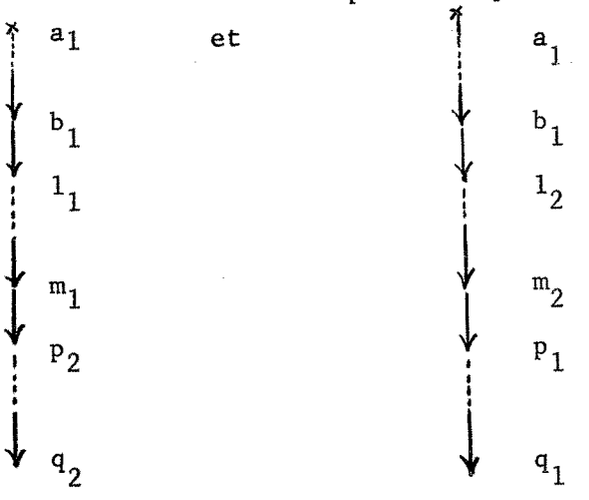
Sachant qu'il n'y a pas de branches divergentes de P à Q.

Soit K :



une partie connexe du catalogue de carte C_1

C'est qu'il existe dans le catalogue de départ deux telles parties connexes qu'on retrouvera dans E_p ($E_o \subset E_p$) à l'ordre des classes près :



Par hypothèse, dans C_p, aucune des classes L ... M n'est comparable avec une des classes P ... Q. La partie connexe K fait donc partie du catalogue final à l'ordre des classes près.

C.Q.F.D.

Théorème 3

On peut décomposer toute transformation en un produit de transformations élémentaires.

Démonstration

Il nous suffit en effet de nous trouver dans les conditions d'application de la propriété 4.

Soit la transformation T_{op}, de C_o à C_p

Nous raisonnons par récurrence.

a) Soit A la première classe de C_p.

Nous faisons passer A à la racine de C_o par le processus P

-On n'a utilisé que des transformations TE1, TE2, TE3.

b) Supposons qu'on ait pu attacher les n premières classes de C_p à leurs gouvernants.

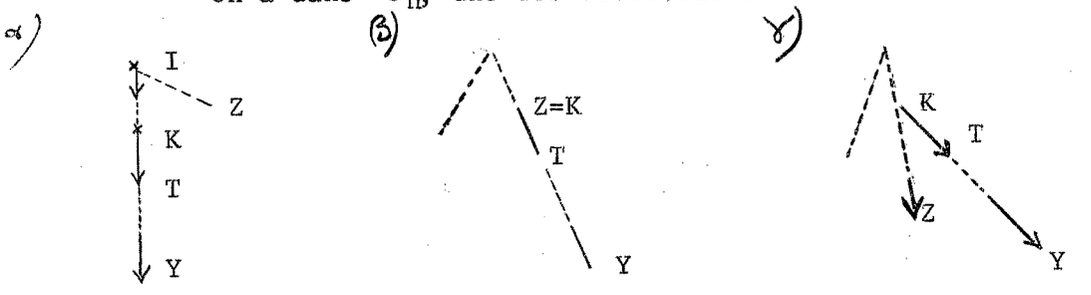
On est arrivé à la carte C_n.

Hypothèse de récurrence

On n'a détruit la comparabilité d'aucun couple de classes comparables dans C_p.

Reportons nous aux définitions de Z, T, K, et Y du théorème 1.

On a dans C_n, une des trois situations suivantes



α) Le cas α) peut se produire. On peut supposer qu'on a laissé T attaché à K en construisant les n premières classes.
Le processus P remonte Y devant T et une TE4 attache à I puis une TE3 à Z, la nouvelle arborescence de racine Y.

Nous sommes sûrs que dans C_p, aucune des classes I ... K n'est comparable avec une des classes T ... Y, les n premières classes étant déjà rangées.

β) Le processus P remonte Y devant T et, éventuellement, une TE2 change l'ordre des dépendants de Z.

γ) Si des classes appartenant à $\hat{\Gamma}(Y)$ dans C_n n'appartiennent pas à $\hat{\Gamma}(Z)$ dans C_p, on les remonte

devant T par le processus P. L'arborescence de racine Y_j débarrassée de ces classes, est transformée par des TE3 pour devenir une branche simple, sans divergences, puis est rattachée à K par une TE4 et à Z par TE3.

c) Finalement nous sommes passés de C_n à C_{n+1} par un produit de transformations élémentaires sans détruire la comparabilité de deux classes comparables dans C_p.

On passe donc de C_o à C_p en conservant cette propriété.

D'après la propriété 4, nous avons bien réalisé la transformation T_{op}.

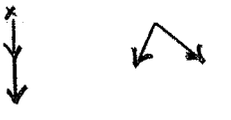
C.Q.F.D.

Nombre de transformations possibles - Catalogue invariant

Cherchons d'abord le nombre de cartes possibles, pour un certain ordre des N classes.

N = 2 une seule carte

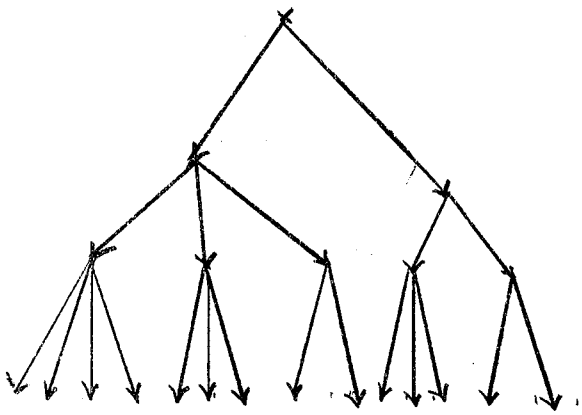
N = 3 deux cartes



N = 4 cinq cartes

Pour ajouter une N + 1ème classe à celles déjà rangées, sachant que la dernière de celles-ci est au niveau x, il y a x possibilités.

Le nombre de cartes possibles pour un certain ordre des N classes est le nombre de noeuds, au niveau N - 1, de l'arborescence suivante :



N = 2 A_2 = 1 carte

N = 3 A_3 = 2 cartes

N = 4 A_4 = 5 cartes

N = 5 A_5 = 14 "

On peut aussi compter A_n avec la matrice B suivante :

$B_{i,1}=i$; $B_{i,j}=0$ si $j>i$; $B_{i,j}= B_{i-1,j} + B_{i,j-1}$

$A_N = B_{N-1, N-1}$

A partir de N=8, on a environ $A_N = 3,5^{N-3}$.

En tout, pour N classes, il y a $(N ! A_N)^2$ transformations possibles.

A partir d'un catalogue K, on peut trouver un catalogue K' dont le contenu est invariant pour n'importe quelle transformation.

Théorème 4

Une condition nécessaire et suffisante pour qu'un produit de transformations conduisant d'une carte C_o à une carte C_p soit une opération interne et associative dans l'ensemble des transformations, est qu'aucune de ces transformations ne détruise la comparabilité de deux classes comparables dans C_p.

En effet, d'après le théorème 3 on peut décomposer ces transformations en transformations élémentaires. L'opération est alors interne par la propriété 4.

-III- ALGORITHMES DES PROGRAMMES DE TRANSFORMATIONS ET DE MODIFICATIONS

Nous avons vu que ces programmes étaient à deux niveaux.

a) Des procédures dites d'entrée sortie, et dépendantes du format de bandes ouvrent et ferment fichiers et catalogues, lisent et écrivent des data.

Ces sous-programmes ont été écrits à la Rand Corporation (9). en M A P 7044.

b) Des programmes utilisant les premiers, et traitant des transformations, correspondant à un changement de carte, et des modifications correspondant à une mise à jour de catalogue. Ils sont aussi écrits en M A P 7044.

Nous avons évité la lecture arrière de data. Ces derniers ne formant pas nécessairement un enregistrement physique, une telle opération nécessite.

a) un repositionnement en arrière .
b) une lecture avant pour chercher le début du datum.
c) éventuellement un nouveau repositionnement en arrière et de nouvelles lectures avant.

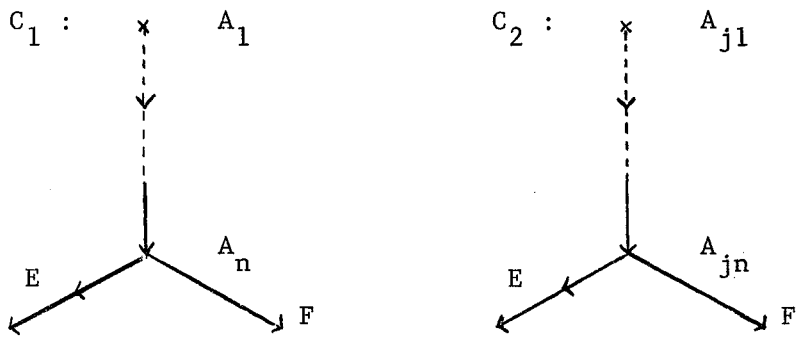
Nous analyserons les transformations TE1, TE2, TE3 et le programme de modifications.

A - TRANSFORMATION TE1

Nous la réaliserons dans le cas général en trois programmes indépendants

1) Un programme de coupure appelé C. TRA2 .
2) Un programme de changement d'ordre des classes appelé C. TRA1 .
3) Un programme de composition appelé C. CØPØ .

Supposons en effet avoir à effectuer la transformation T_{12}



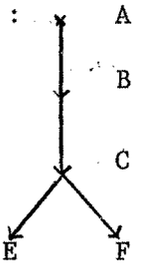
Dans ce programme, nous avons à changer l'ordre des classes de A_1 à A_n pour chaque suite a_{1i}, ... a_{ni} et à trier suivant le critère d'ordre correspondant à C_2 :

Il est plus commode, pour pouvoir comparer a_{j1}, ... a_{jn} à a'_{j1} ... a'_{jn} que tous ces data tiennent ensemble en mémoire centrale.

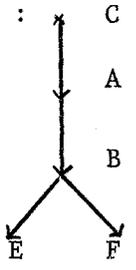
Ceci sera difficilement réalisé si ces deux sous entrées sont accompagnées de $\hat{\Gamma}(a_{jn})$ et $\hat{\Gamma}(a'_{jn})$. Nous couperons donc C_1 en deux parties que nous rattacherons dans C. CØPØ.

1) Le programme de coupure (C. TRA2)

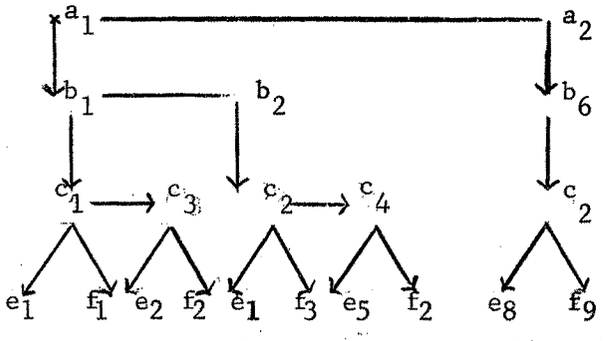
On coupe ici le catalogue en deux : chaque structure attachée à a_{ni}, de la classe A_n, est placée sur une bande auxiliaire avec un numéro provisoire représentant le numéro d'occurrence de a_{ni} dans le catalogue. Ainsi, soit la transformation: C_1 :



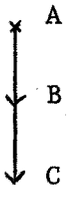
C_2 :



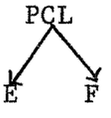
et le catalogue



On le sépare en deux catalogues, l'un de carte

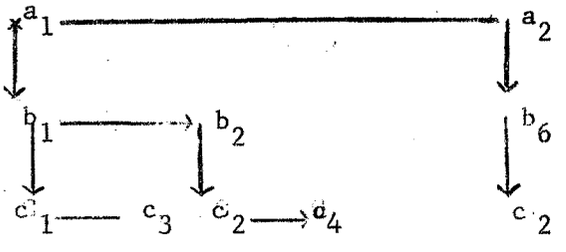


et l'autre de carte

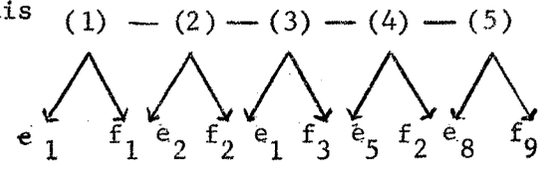


PCL étant une classe où tous les data ont une longueur d'une mémoire et contiennent un numéro.

On aura :



puis



Paramètres de ce programme

Supposons que : NØM1 soit le nom du catalogue de départ, EDIT1 son numéro d'édition,

NØM2 celui du second catalogue et EDIT2 celui de son numéro d'édition,

NØM3 et EDIT3 respectivement celui du 3ème catalogue et de son numéro d'édition.

- C est la classe sur laquelle seront pris les numéros provisoires nommés NUA.
- E et F, les classes extrêmes de la structure à détacher.
- On appellera le programme par l'instruction = CALL C. TRA2 (NØM1, EDIT1, C, E, F, NØM2, EDIT2, NØM3, EDIT3)

Algorithme

- Nous devons d'abord créer les cartes des catalogues NØM2 et NØM3 en fonction de la carte de NØM1.

Puis nous lisons NØM1

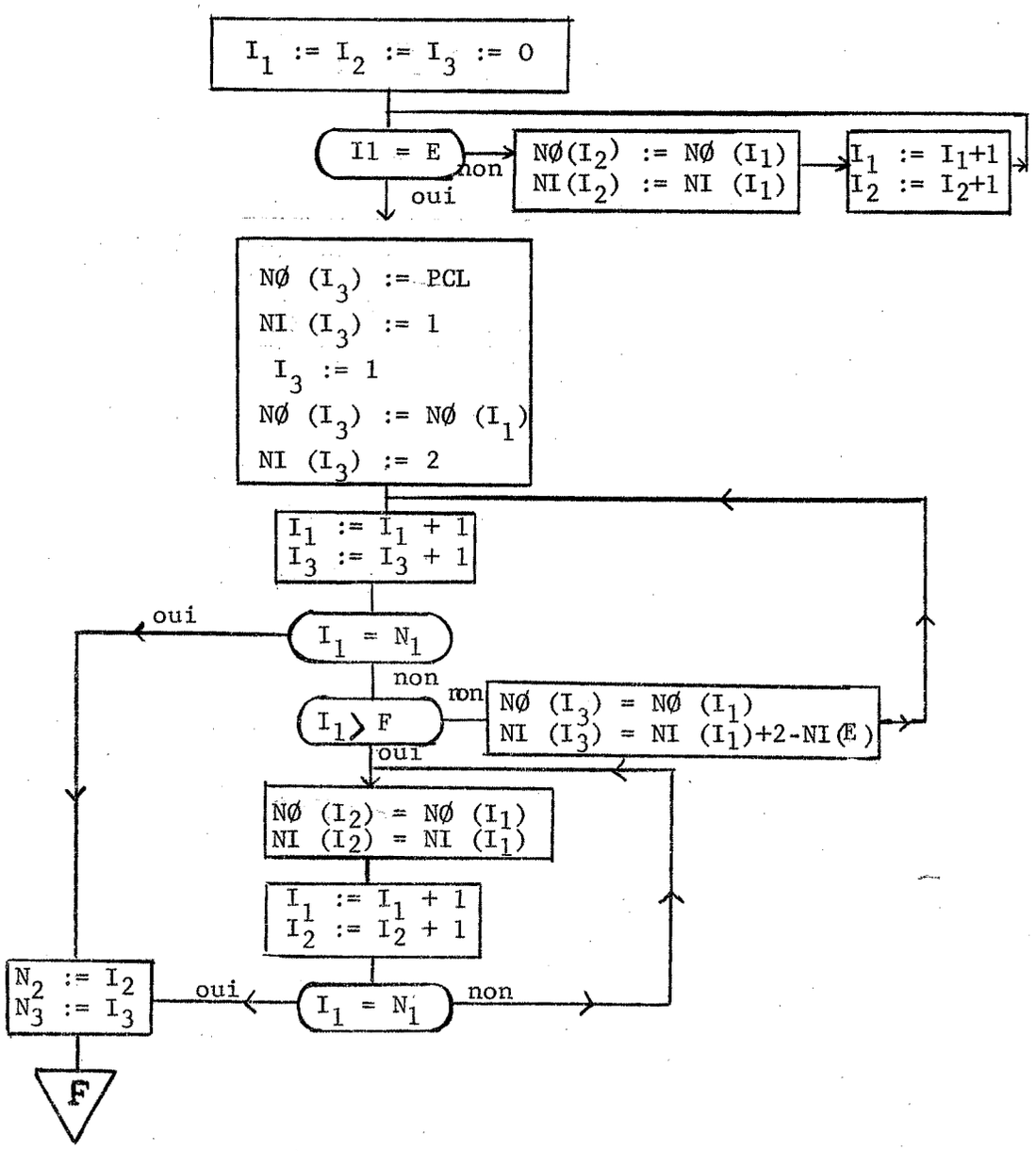
- NUA est le numéro d'occurrence de C. On l'initialise à zéro.
- A chaque rencontre d'un datum de classe E, précédé d'un datum de classe C, NUA devient NUA + 1. On écrit sur NØM3 ce numéro NUA (classe PCL) puis le datum rencontré.

Les data qui suivent, s'ils sont de classe comprise entre E et F vont sur NØM3, sinon sur NØM2.

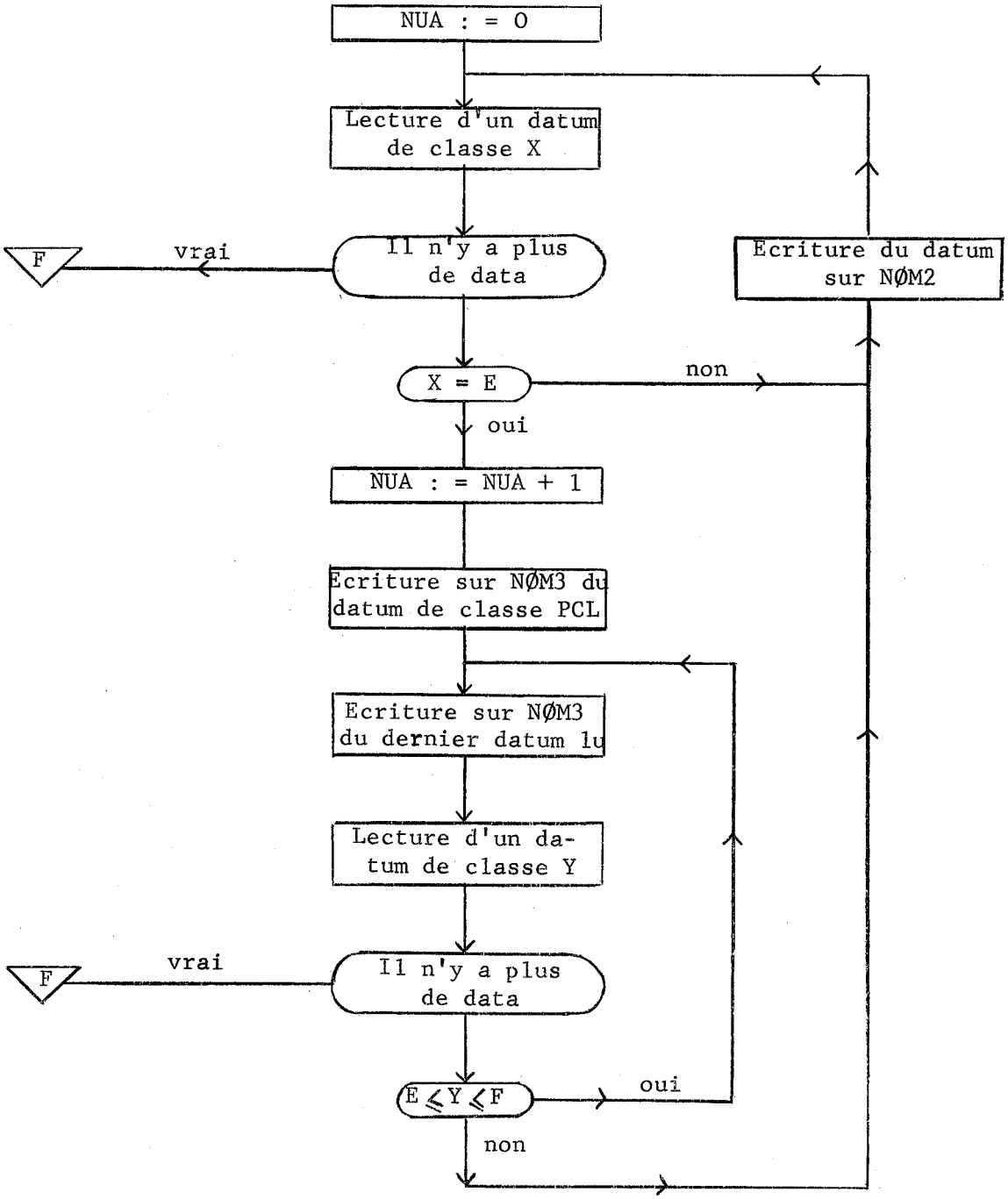
Organigramme de création des cartes.

I_1 est le numéro de classe de la carte de NØM1 qui a N_1 classes
I_2 " " " " " NØM2 " " N_2 "
I_3 " " " " " NØM3 " " N_3 "

Le nom de la classe I sera appelé NØ(I), son niveau sera NI(I).

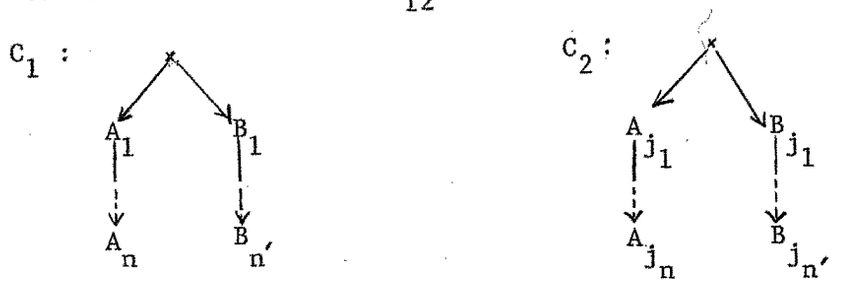


Organigramme de C. TRA2



2) Le programme de changement d'ordre des classes (C.TRA1)

Soit la transformation T_{12} suivante

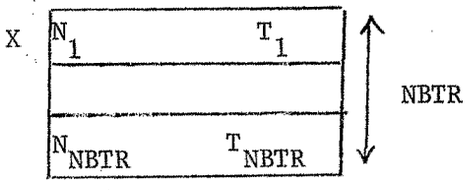


C.TRA1 permet la réalisation simultanée des deux transformations TE1.

a) Paramètres de ce programme

On part d'un catalogue NØM1, de numéro d'édition EDIT1 pour arriver au catalogue NØM2, EDIT2.

C.TRA1 effectue NBTR transformations TE1 décrites à partir d'une mémoire X.



La mémoire T_i (T_1, T_2 ...) contient en partie adresse le numéro de classe de la racine de l'arborescence à transformer (dans l'exemple, les numéros des classes A_1 et B_1).

Si $N_i = 0$ C.TRA1 supprimera totalement cette arborescence.

On note que ces arborescences étaient au départ de simples branches ou le sont devenues après exécution de C.TRA2. Le traitement de ce dernier cas est un peu différent puisqu'il faudra après C.TRA1 rattacher des structures à A_{j_n} et $B_{j_n'}$ dont le numéro d'occurrence n'est plus le même dans NØM2 que dans NØM1. Un comptage sera alors à effectuer.

L'utilisateur préviendra C.TRA1 en signant - la mémoire T_i correspondante. En cas de non suppression, on ordonne dans les N_i (N_1,N_2...) mémoires qui suivent T_i les classes de la nouvelle branche. N_i peut d'ailleurs être inférieur au nombre de classes de l'ancienne branche et dans ce cas, C.TRA1 en supprime une ou plusieurs.

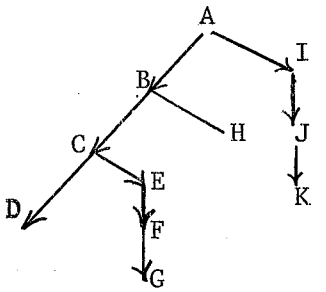
En partie décrément de ces N_i mémoires, on peut, si on désire une vérification, mettre le futur numéro de niveau de ces classes.

L'appel sera :

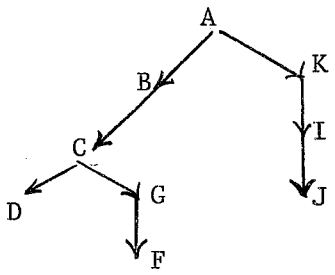
CALL C.TRA1 (NØM1, EDIT1, NBTR, X, NØM2, EDIT2)

Exemple :

C_1 :



C_2 :



Numéro des classes : A : 0 B : 1 C : 2 D : 3 E : 4 F : 5
G : 6 H : 7 I : 8 j : 9 K : 10

On devra trouver le nombre 3 dans la mémoire NBTR, puis

X :	2	T_1
X+1	0	T_2
X+2	3	T_3

T_1 :

4
6
5

2

ou

T_1 :

	4
4	6
5	5

T_2 :

7

T_3 :

8
10
8
9

3

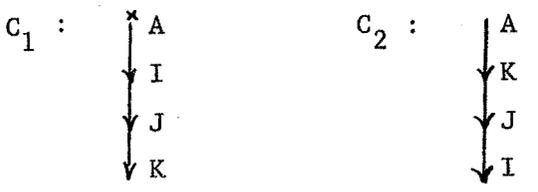
ou

	8
2	10
3	8
4	9

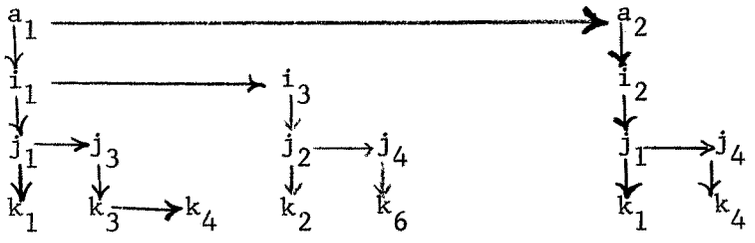
C.TRA1 lira pour T_1 par exemple : l'ancienne branche de racine 4 devient une branche dont les classes sont 6 et 5.

b) Algorithme

Nous allons le voir sur un exemple simple.



soit le catalogue :



Une première manière de procéder pourrait être de changer l'ordre I J K en K J I pour chaque triplet i j k, puis de renvoyer le nouveau triplet sur le catalogue de sortie. Il ne resterait plus alors qu'à trier le catalogue entier.

L'inconvénient de cette méthode est que les data de classe A vont être inutilement triés.

Nous décidons donc de trier immédiatement les triplets k j i et de les écrire dans le bon ordre sur le catalogue de sortie.

2 cas se présentent

1°) Tous les data de i_1 à k_6 tiennent en mémoire

Nous montrons comment on occupe la mémoire centrale.

Nous avons :

- une table U de trois mémoires par élément où se trouvent les adresses des data k, j, i d'un triplet et leur longueur en décrément.

- une table V d'une mémoire par élément, contenant l'adresse d'un élément de U.

- enfin les data eux-mêmes qui sont rangés dans l'ordre où ils sont lus.

Les éléments de U sont rangés dans l'ordre où les data, auxquels ils font allusion, sont lus.

Les éléments de V sont rangés dans l'ordre du nouveau classement.

Ainsi on aurait, en supposant les data longs de une mémoire :

Adresse de la mémoire	Son contenu		
1		7	V
2		I6	
3		I0	
4		I3	
5		I9	
6			
7	I	52	U
8	I	5I	
9	I	50	
I0	I	54	
II	I	53	
I2	I	50	
I3	I	55	
I4	I	53	
I5	I	50	
I6	I	58	
I7	I	57	
I8	I	56	
I9	I	60	
20	I	59	
2I	I	56	

data

Adresse de la mémoire	Son contenu
50	i_1
5I	j_1
52	k_1
53	j_3
54	k_3
55	k_4
56	i_3
57	j_2
58	k_2
59	j_4
60	k_6

Dans un problème réel, évidemment, la liste des data est plus longue que la table U.

2°) Tous les data, de i_1 à k_6 ne tiennent pas en mémoire centrale :

Supposons qu'il n'y ait plus de place pour le premier k_4.

On a donc seulement dans V :

1	7
2	I0
3	0

La table U est remplie jusqu'à la mémoire 12.
Les data vont jusqu'à 54.
Dans ce cas on vide la mémoire centrale sur la bande auxiliaire AUX1. On change alors la place des data i_1 ,j_3, k_3 qu'on garde au début de la zone libre de la mémoire.
On aura :

1	0		V
7		52	U
8		51	
9		50	

50	i_1
51	j_3
52	k_3

Sur AUX1 on a : $k_1 j_1 i_1 \quad k_3 j_3 i_1$.
On lit maintenant $k_4 i_3 j_2 k_2$ par exemple.

Sur une seconde bande auxiliaire AUX2, on mettra $k_2 j_2 i_3 k_4 j_3 i_1$.

Enfin on lit $j_4 k_6$. La sous-entrée $k_6 j_4 i_3$ étant supérieure à $k_4 j_3 i_1$, il n'est pas nécessaire de changer à nouveau de bande. On écrit $k_6 j_4 i_3$ sur AUX2.
AUX1 : $k_1 j_1 i_1 \quad k_3 j_3 i_1$.
AUX2 : $k_2 j_2 i_3 \quad k_4 j_3 i_1 \quad k_6 j_4 i_3$.

Un interclassement de AUX1 et AUX2 donne alors les cinq triplets dans le bon ordre. On peut évidemment avoir besoin de plusieurs passages de bande pour l'interclassement, ce qui n'est pas le cas dans l'exemple précédent.

L'interclassement sera terminé quand tout sera arrivé sur une même bande. Il n'y a plus alors qu'à transférer le contenu de cette bande sur la bande normale de sortie.

En continuant à lire la bande d'entrée, les mêmes événements se représenteront après la lecture de a_2.

Quand ce programme est réalisé après la phase C.TRA2, il faut ajouter plusieurs tris pour retrouver les bons numéros dans la classe PCL.

Pour cela, dans C.TRA1 nous faisons accompagner chaque datum k de son ancien numéro. Nous pouvons alors trouver un catalogue à deux classes : Nouveau Numéro NN et Ancien Numéro AN.

$\downarrow^{NN}_{AN}$ nous transformons ce catalogue en un catalogue de carte : $\downarrow^{AN}_{NN}$

On peut alors remplacer dans la classe PCL les anciens numéros par les nouveaux et trier le catalogue de première classe PCL.

c) Sous-programmes utilisés

Le sous-programme DRAN permet de référencer un datum dans la zone U.

Le sous-programme RANGE met à jour la table V et l'ordonne.

Sachant qu'on lit sur bande i_1, j_1, k_1, j_3, k_3, le sous-programme CØMPLE, à la lecture de j_3, k_3 restitue à la table U l'adresse de i_1.

C.CØMP compare des data sur un critère de longueur ou d'ordre alphabétique.

C.CØEN compare des sous-entrées i j k.

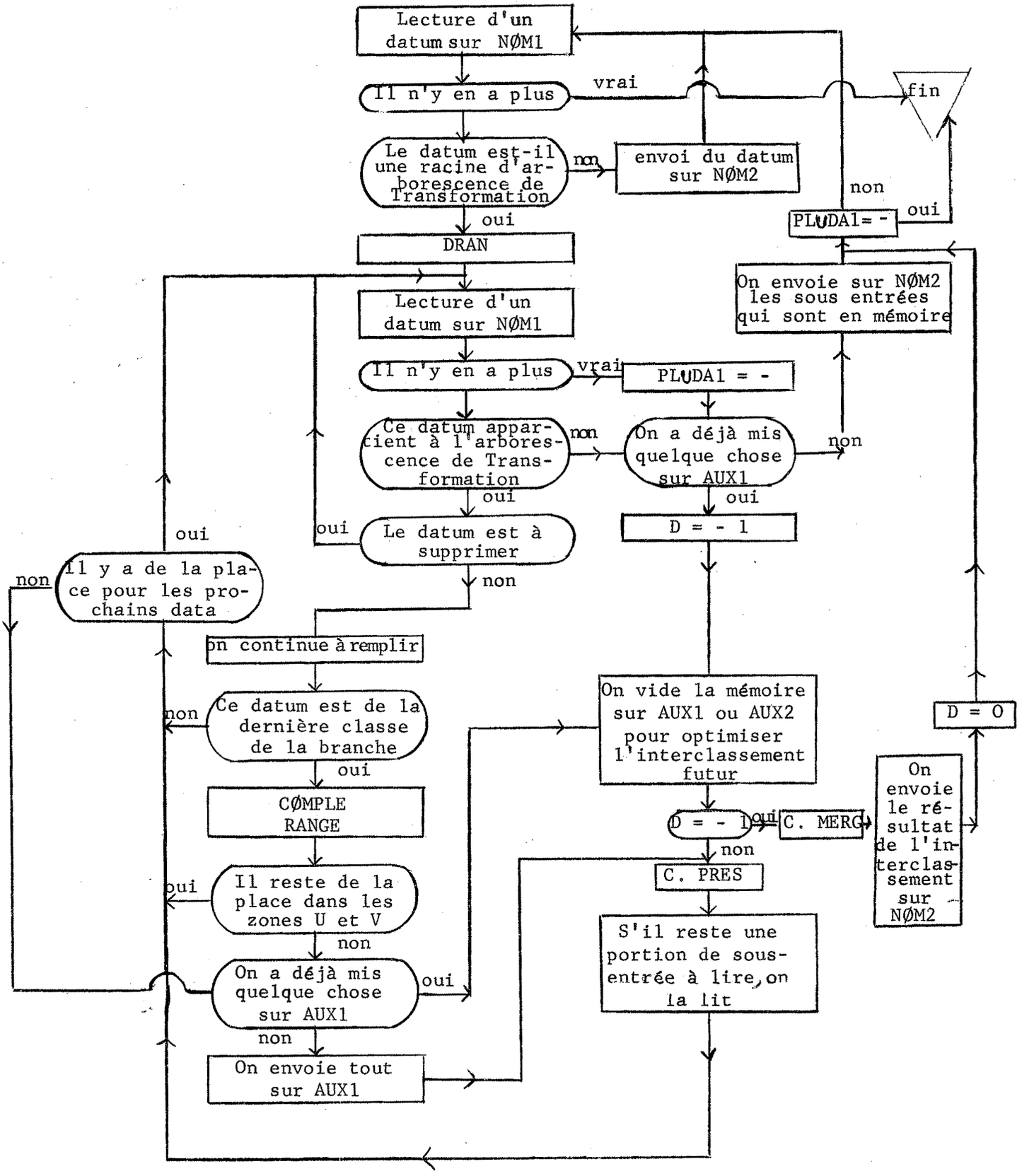
C.LENT et ECRBEN font correspondre la table V et la bande par un jeu d'adressage indirect.

C.MERG interclasse les 2 bandes AUX1 et AUX2 en se servant des deux autres AUX3 et AUX4.

C.PRES ramène un ou plusieurs triplets au début de la mémoire centrale.

Nous donnons l'organigramme correspondant au cas où l'on n'utilise pas C.TRA1 à la suite de C.TRA2.

d) Organigramme



3) Programme de composition (C.CØPØ)

Nous avons coupé au 1) le catalogue de départ en deux catalogues. C'est le moment maintenant de les rassembler.

Supposons qu'il s'agisse d'attacher à la classe A , derrière la classe D du catalogue NØM1, EDIT1, les structures du catalogue NØM2, EDIT2. La classe 0 (PCL) du catalogue NØM2 représente l'ensemble des numéros provisoires de la classe A de NØM1. Le catalogue obtenu sera NØM3, EDIT3.

On appelle ce sous-programme par :

CALL C.CØPØ (NØM1, EDIT1, NØM2, EDIT2, A, D, NØM3, EDIT3)

C.CØPØ fait en plus une vérification des numéros de la classe PCL.

Les cas D = A et D ≠ A sont toujours à différencier car si D = A le rattachement est fait juste après les data a , sinon sous a, après tous les d, ce qui justifie qu'on attende d'avoir lu un d puis un datum autre que d pour rattacher entre ces deux derniers data. La condition est :

$$\text{ØVDL1D} = -1 \wedge \text{ØL1D} = +1$$

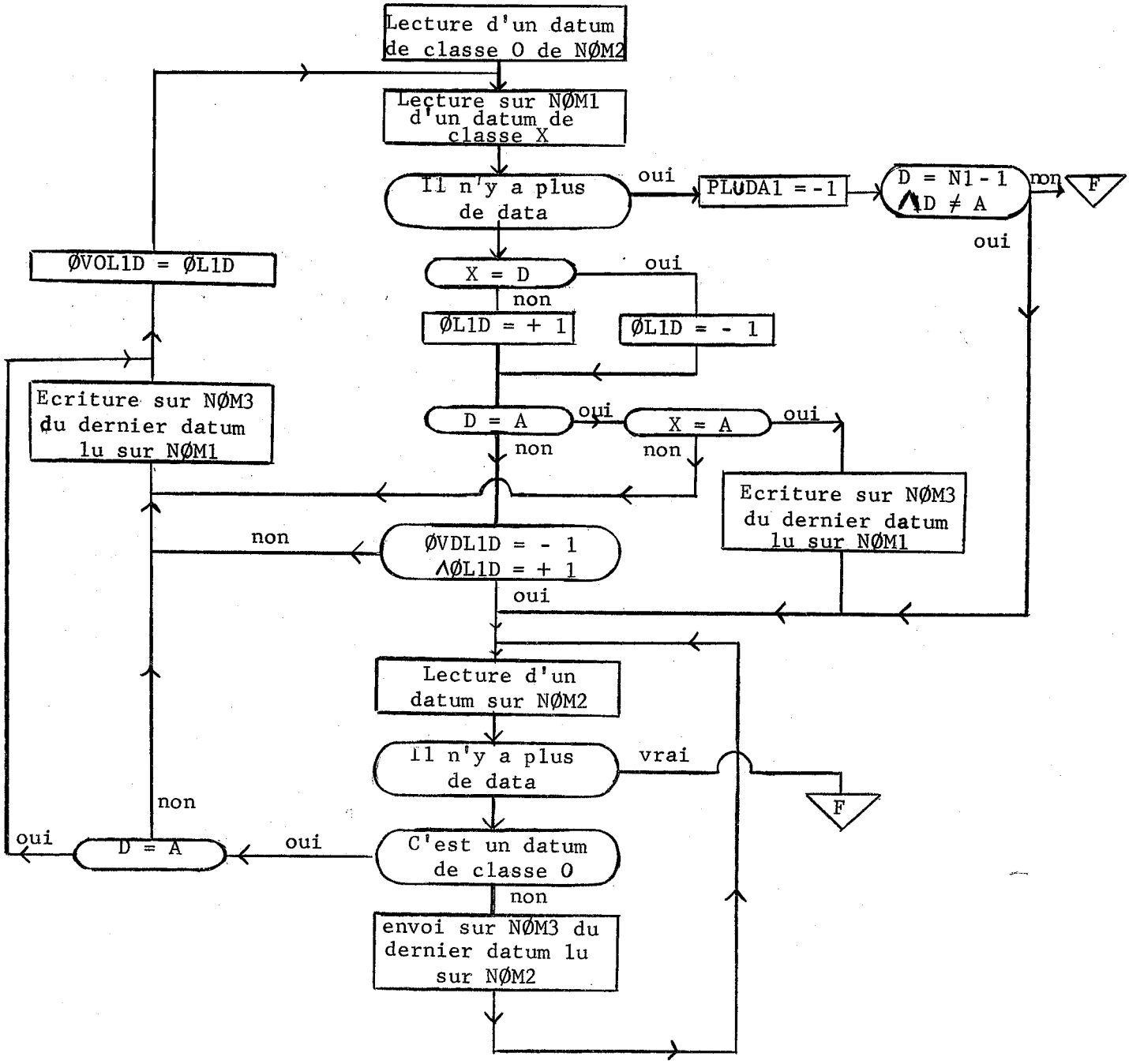
Nous donnons l'organigramme à la page suivante.

- TRANSFORMATION TE2

Celle-ci se réalise en deux programmes déjà étudiés : le programme de coupure et celui de composition.

Organigramme

La création de la carte de NØM3 à partir des cartes de NØM1 et NØM2 est l'opération inverse de la séparation. Voici l'organigramme de C. CØPØ:

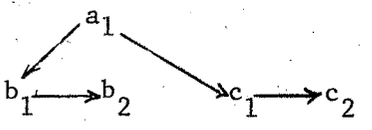


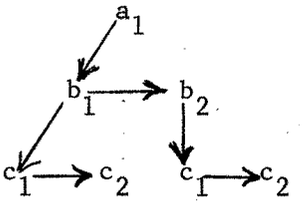
C - TRANSFORMATION TE3 (C.TRA3 et C.COPØ)

Cette transformation se réalise en deux parties : une coupure et une composition. Mais le programme de coupure diffère notablement de celui que nous avons déjà étudié.

1) Divers algorithmes possibles

a) Supposons en effet que nous fassions une coupure ordinaire : la composition va s'avérer délicate car il faut associer tous les c à chaque b.

Soit : 

devenant 

c_1 - c_2 pouvant représenter une structure très grosse, qui ne tient pas en mémoire centrale, il faudrait après avoir écrit c_1 - c_2 sous b_1, le relire et l'écrire à nouveau sous b_2. Nous avons déjà vu les difficultés de la lecture arrière et ne pouvons donc envisager cette solution.

b) Nous pourrions alors espérer que si c_1 - c_2 est une grosse structure, c_1 ou c_2 représentent des objets plus petits. Nous pourrions alors, à la lecture de la bande d'entrée créer :

(1) → (2) → (1) — (2)
↓ ↓ ↓ ↓
c_1 c_2 c_2 c_2

Les numéro (1) et (2) étant les numéros d'occurrence de b_1 et b_2 Nous trierions alors en fonction des numéros provisoires puis des data et obtiendrions :

(1) → (1) → (2) → (2)
↓ ↓ ↓ ↓
c_1 c_2 c_1 c_2

Nous pouvons, en fait, éviter ce tri de la manière suivante :

c) Si la structure $c_1 - c_2$ tient en mémoire centrale, nous mettons sur la seconde bande de sortie NØM3 :

$$\overset{(1)}{\underset{\downarrow}{c_1}} \rightarrow c_2 \qquad \overset{(2)}{\underset{\downarrow}{c_1}} \rightarrow c_2$$

sinon nous vidons la mémoire centrale sur une bande auxiliaire AUX1.

On continue à lire NØM1, et écrire sur AUX1 et sur NØM3

On a créé $\overset{(1)}{\underset{\downarrow}{c_1}} \rightarrow c_2$ sur NØM3. Il suffit de rebobiner AUX1 et de le relire pour mettre à la suite dans NØM3 : $\overset{(2)}{\underset{\downarrow}{c_1}} \rightarrow c_2$

On recommencera ce processus autant de fois qu'il y a de b_i, moins AUX1 sert seulement de temps à autre, et quand il sert, on a besoin d'un rebobinage rapide. Une technologie à disques est donc tout à fait indiquée dans ce cas.

2) Paramètres de C.TRA3

Le catalogue de départ sera NØM1, EDIT1 et les catalogues à obtenir NØM2, EDIT2 puis NØM3, EDIT3.

Il s'agit de couper de A l'arborescence de racine C pour la rattacher plus tard à B.

C.TRA3 permet aussi de supprimer n arborescences de racines S_1, S_2. ...S_n (numéros de classes). On a n dans une mémoire S et S_1,S_2...S_n dans les mémoires qui suivent S.

L'appel sera :

CALL C.TRA3 (NØM1, EDIT1, B, C, S, NØM2, EDIT2 NØM3, EDIT3).

3) Sous-programmes utilisés

CREMAP crée les cartes de NØM2 et NØM3 à partir de la carte de NØM1. Son organigramme a déjà été vu pour C.TRA2.

APPSUP décide si telle classe est à supprimer du catalogue.

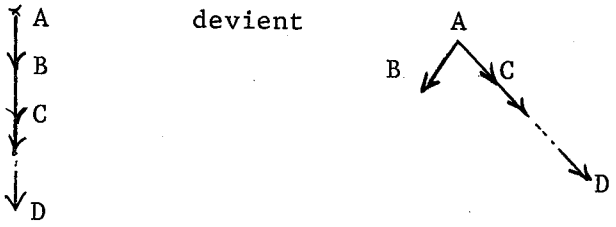
STØCKE va stocker la mémoire centrale sur NØM3 ou sur AUX1.

4) Organigramme

Nous le donnons à la page suivante.

- TRANSFORMATION TE4

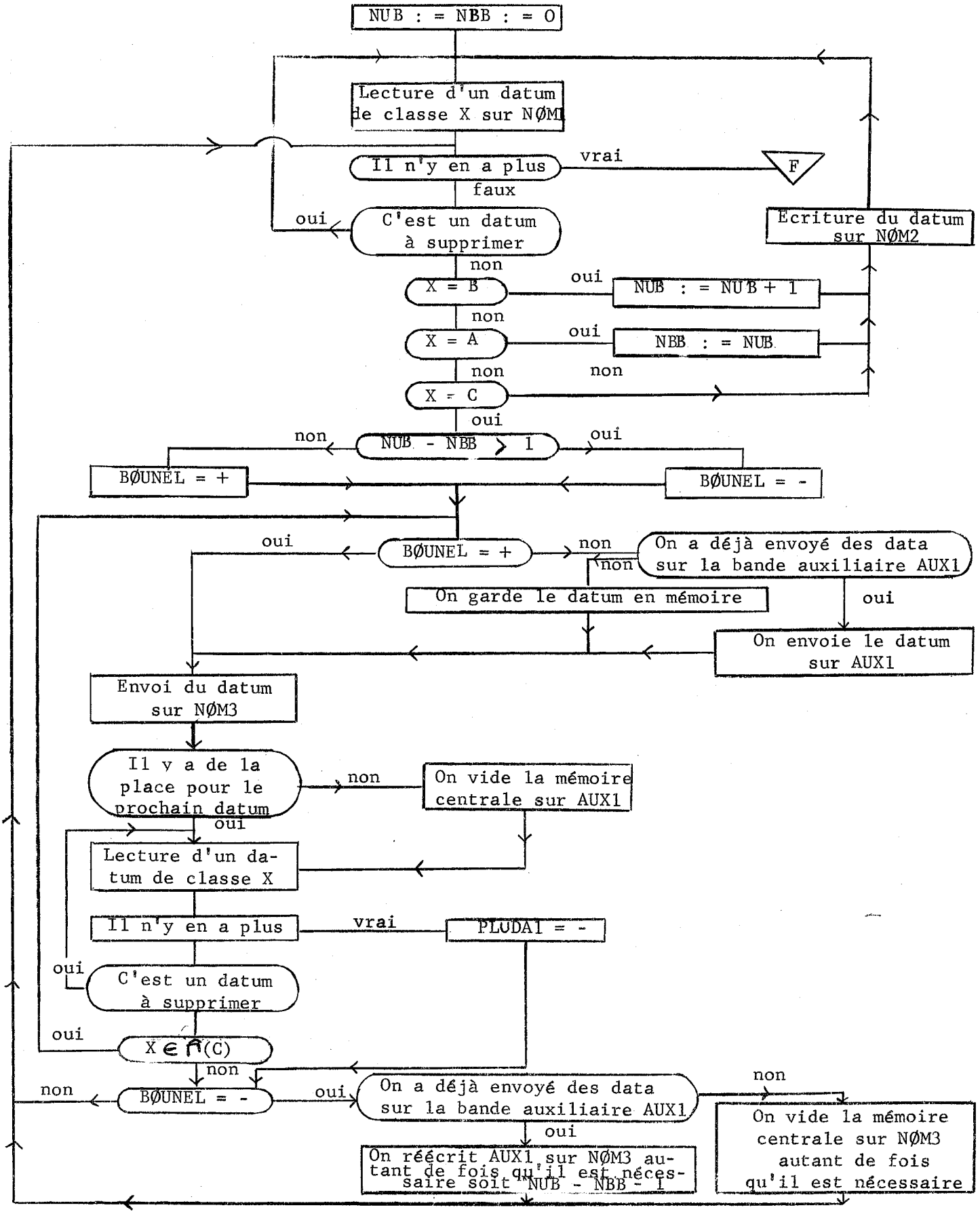
Ce programme n'a pas été réalisé. Toutefois, nous disposons de suffisamment de sous-programmes pour le faire.



Un programme de séparation nous permet de couper la branche qui va de C à D par C.TRA2.

Il faut de plus trier les diverses branches, ce que nous avons réalisé dans C.TRA1.

Puis on doit attacher ces branches à A par C.CØPØ.


NUB : = NBB : = 0
Lecture d'un datum de classe X sur NØM1
Il n'y en a plus
vrai
F
faux
oui
C'est un datum à supprimer
Ecriture du datum sur NØM2
non
X = B
oui
NUB : = NUB + 1
non
X = A
oui
NBB : = NUB
non
non
X = C
oui
non
NUB - NBB > 1
oui
BØUNEL = +
BØUNEL = -
oui
BØUNEL = +
non
On a déjà envoyé des data sur la bande auxiliaire AUX1
non
On garde le datum en mémoire
oui
On envoie le datum sur AUX1
Envoi du datum sur NØM3
Il y a de la place pour le prochain datum
non
On vide la mémoire centrale sur AUX1
oui
Lecture d'un datum de classe X
Il n'y en a plus
vrai
PLUDA1 = -
oui
C'est un datum à supprimer
oui
X ∈ Π(C)
non
non
BØUNEL = -
oui
On a déjà envoyé des data sur la bande auxiliaire AUX1
non
oui
On réécrit AUX1 sur NØM3 autant de fois qu'il est nécessaire soit NUB - NBB - 1
On vide la mémoire centrale sur NØM3 autant de fois qu'il est nécessaire

- ETUDE DES MODIFICATIONS

Il s'agit ici, de changer certains data d'un catalogue en gardant la même carte.

Il y a un grand nombre de modifications possibles. Nous en avons choisi quatre :

- Addition
- Suppression
- Deletion
- Remplacement.

1) Définition

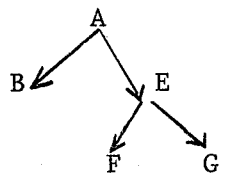
a) Addition

Il s'agit d'ajouter au catalogue un datum α et un ensemble de descendants de α .

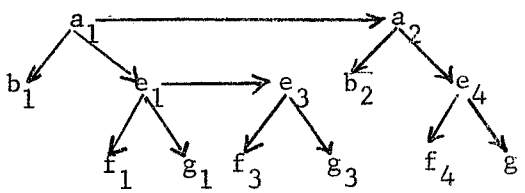
Le contexte : Il faut évidemment préciser à quels endroits du catalogue nous désirons placer α . Nous donnons alors le contexte de α qui est une suite de data dont chacun, excepté le premier, est, dans le catalogue, un descendant du précédent.

Exemple : Nous désirons ajouter le datum e_2 suivi des data f_2, f_3, g_2 dans le catalogue NØM1, de carte C_1

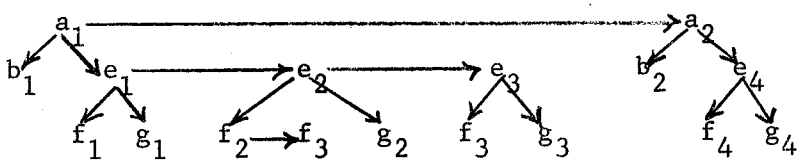
C_1 :



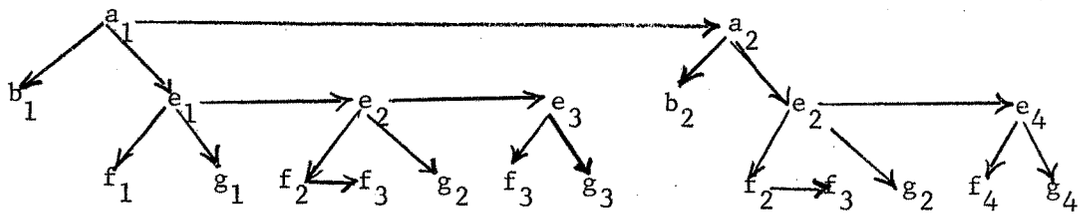
NØM1 :



si nous précisons le contexte a_1 par exemple, la modification donne le catalogu



Si on ne précise aucun contexte, on aura :



Le contexte est laissé au choix de l'utilisateur.

b) Suppression

Il s'agit de supprimer un datum d'un catalogue, dans un contexte donné ou non.

c) Deletion

On supprime un datum α et tous les data de l'arborescence de racine α.

On remplace en réalité tous ces data par des data nuls.

d) Remplacement

On remplace un datum par un autre datum.

Dans les divers contextes, on peut faire allusion à un datum qu'on va supprimer ou remplacer, ou à tous autres data, excepté ceux qu'on va ajouter délaisser ou par lesquels on va remplacer des data.

2) Paramètres du programme de modifications

Le programme permet de faire un nombre quelconque d'additions, suppressions, deletions, remplacements.

Toutefois, il a été réalisé en supposant que tous les data auxquels on fait allusion sont en mémoire centrale, alors que le catalogue est sur bande. Le support physique de ces data peut être une bande ou un paquet de cartes, mais il faut les lire avant d'appeler le programme de modifications.

Cette restriction peut être ennuyeuse pour des additions. Si nous devions nous trouver en présence de problèmes trop gros, nous pourrions modifier légèrement le programme pour surmonter la difficulté en lisant les data à additionner, sur bande, au lieu de les trouver directement en mémoire.

Finalement, si nous désirons modifier un catalogue NØM1, EDIT1 pour en faire un catalogue NØM2, EDIT2 on appelle le programme par :
CALL C.MØDI (NØM1, EDIT1, DESCR, NØM2, EDIT2).

Dans la mémoire DESCR se trouve en partie adresse, l'adresse A de la description des additions, en décrément le nombre NA d'additions.

En DESCR + 1 se trouvent S, adresse de la description des suppressions et NS.
En DESCR + 2 , on a D et ND.
En DESCR + 3 , on a R et NR.
En A et dans les NA-1 mémoires qui suivent, on trouve en adresse, l'adresse de la description R_i d'une addition particulière, en décrément, le nombre N_i de mémoires de cette description.

De R_i à R_i+N_i-1 on a dans l'ordre des classes, en adresse, l'adresse de référence des data du contexte, en décrément leur classe, puis les mêmes renseignements pour le premier datum à ajouter.

L'adresse de référence d'un datum est l'adresse d'une mémoire contenant l'adresse et la longueur du datum.

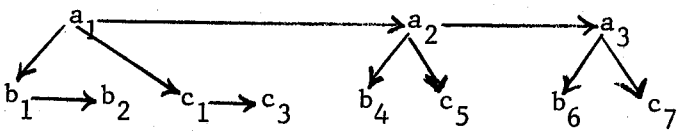
Dans R_i+N_i on a le nombre TA_i de data à ajouter moins 1 puis à la suite les renseignements concernant ces TA_i -1 data.

Pour S et D ce sera la même chose, mais C.MØDI ne trouvera des renseignements que jusque $R_i + N_i - 1$.

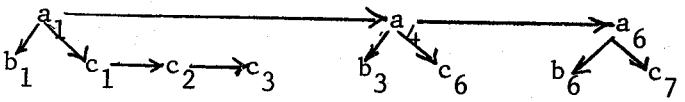
Pour R on aura en $R_i + N_i$ les renseignements concernant le datum de remplacement.

Nous donnons un exemple pour préciser cette séquence d'appel. Nous assimilerons ici le datum à son adresse de référence.

Soit la carte 

et le catalogue 

que nous voulons modifier pour qu'il devienne :



On doit ajouter c_2 dans le contexte a_1

ajouter a_4 b_3 c_6 sans contexte

supprimer b_2 dans le contexte a_1

délaisser a_2 et son arborescence sans contexte

remplacer a_3 par a_6.

A la mémoire DESCR, on aura :

DESCR

2	A
1	S
1	D
1	R

A

2	R_1
1	R_2

S

2	R_3

D

1	R_4

R

1	R_5

R_1

0	a_1
2	c_2
	0

Addition d
seul datum
c_2

R_2

0	a_4
	2
1	b_3
2	c_6

Addition d
a_4, b_3, c

R_3

0	a_1
1	b_2

suppressio
de b_2

R_4

0	a_2

deletion d

R_5

0	a_3
0	a_6

remplacem
de a_3 par

3) Algorithme du programme de modifications

Il nous suffit de savoir pour chaque liste R_i, où on en est dans le catalogue.

Par exemple, pour R_1, il faut savoir si les data qu'on rencontre sont des descendants de a_1 pour décider si on doit ajouter c_2.

Pour chaque liste il nous faut un index qui oscille entre les valeurs 0 et N_i.

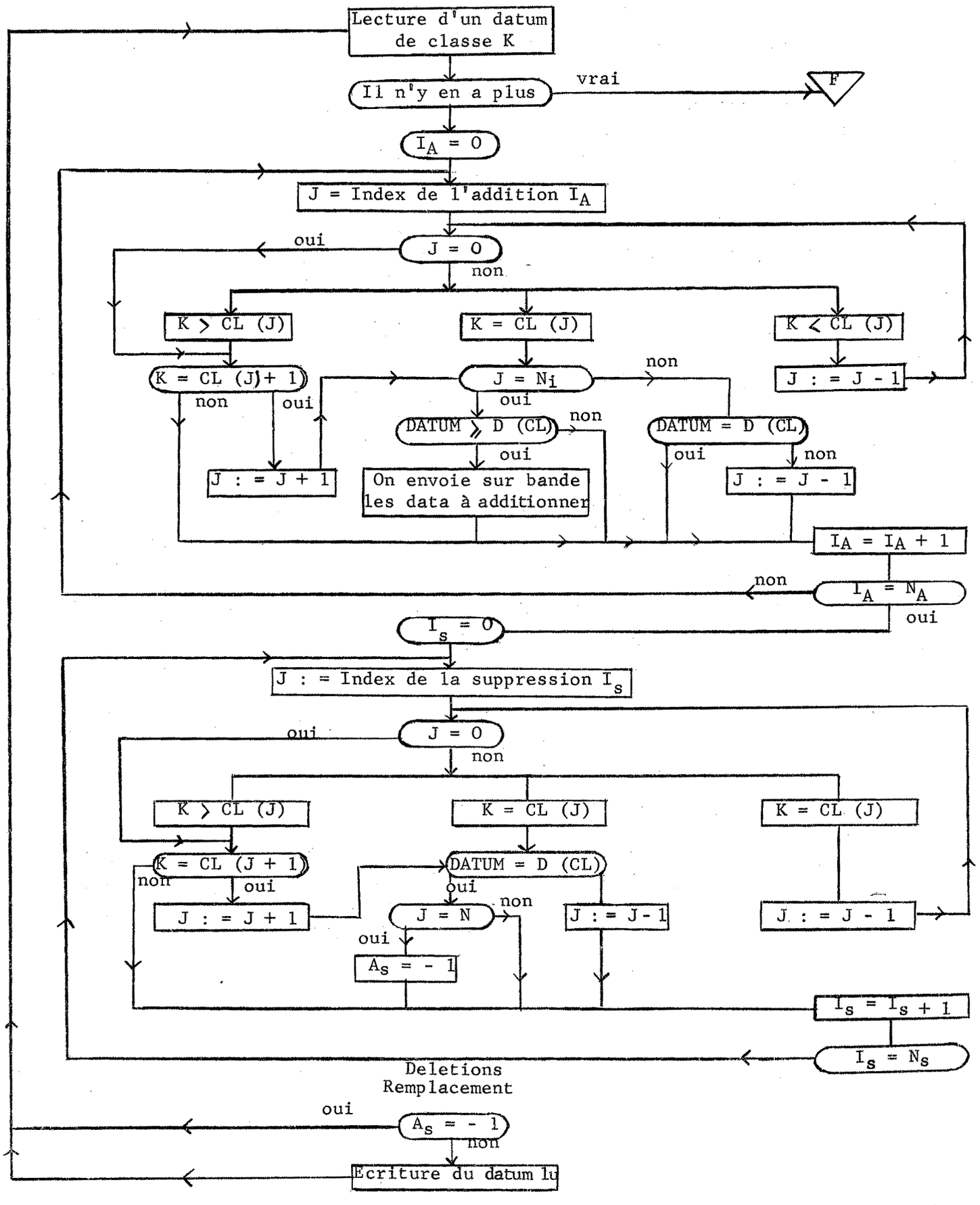
Dans R_1, dès la lecture de a_1, l'index va pointer a_1. A la lecture de c_3, la place de l'index indique qu'on a le bon contexte. Puisque $c_2 < c_3$ on doit donc placer c_2 avant c_3. On s'aperçoit alors qu'il n'y a aucun datum à ajouter derrière c_2. Quand on aura lu a_4, l'index pointera la mémoire qui précède R_1.

L'algorithme consiste donc à tenir à jour une table d'index et, quand l'un d'eux arrive à la valeur N_i, à faire l'opération correspondant à R_i.

L'index peut, en fait, descendre et monter plusieurs fois en cours d'exécution du programme. Ceci revient à dire qu'on peut ajouter, supprimer, délaisser, ou remplacer plusieurs fois les mêmes data, ce qui serait plus délicat si on devait avoir des modifications ne tenant pas en mémoire centrale. Il faudrait alors soit faire des lectures en arrière, soit faire plusieurs passages de bande.

L'organigramme se décompose en quatre phases très semblables. Nous donnons les deux premières phases, celles de l'addition et de la suppression.

J représentera l'index correspondant à une addition ou suppression particulière. D(J) et CL(J) représenteront le datum que l'index pointe et sa classe.


Lecture d'un datum de classe K
Il n'y en a plus
vrai
F
$I_A = 0$
J = Index de l'addition I_A
J = 0
oui
non
K > CL (J)
K = CL (J)
K < CL (J)
K = CL (J)+ 1
non
oui
$J = N_i$
non
oui
J : = J - 1
DATUM ⩾ D (CL)
non
oui
DATUM = D (CL)
oui
non
J : = J + 1
On envoie sur bande les data à additionner
J : = J - 1
$I_A = I_A + 1$
non
$I_A = N_A$
oui
$I_s = 0$
J : = Index de la suppression I_s
J = 0
oui
non
K > CL (J)
K = CL (J)
K = CL (J)
K = CL (J + 1)
non
oui
DATUM = D (CL)
oui
J : = J + 1
J = N
non
J : = J - 1
J : = J - 1
oui
$A_s = -1$
$I_s = I_s + 1$
$I_s = N_s$
Deletions
Remplacement
oui
$A_s = -1$
non
Ecriture du datum lu

IV- APPLICATIONS DES CATALOGUES

· TEXTES SOUS FORMAT DE CATALOGUES

Un ensemble de textes appelé corpus est en train d'être mis sous format de catalogues.

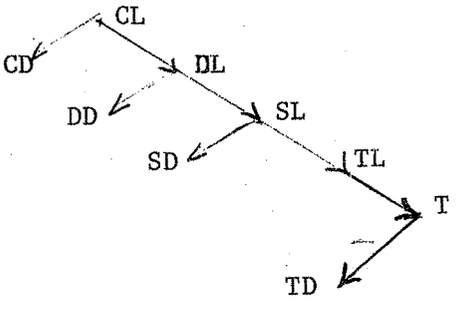
Le corpus était déjà sur bande magnétique. Il a fallu changer son format. Pour cela des programmes dépendants de l'ancien format ont préparé une suite de data pour le programme "Ecrire un datum" du système de catalogues

Le corpus va maintenant être écrit ainsi :

- Les lignes de texte seront les data principaux.
- Chaque ligne sera accompagnée d'informations descriptives.
- Une collection de lignes sera une section accompagnée d'une étiquette et d'u description.
- Une collection de sections sera une division avec étiquette et description.
- Une collection de divisions formera un corpus.

On pourra en réalité trouver plusieurs corpora sur un même catalogue Ce dernier aura alors plusieurs data de classe O.

Finalement la carte sera :



Nous avons à gauche les descriptions des -corpus, divisions, sections- , à droite les étiquettes. Les lignes du texte sont les data de la classe T.

Un enquête récente a permis de montrer que beaucoup de centres de mécano-linguistique ou de traduction automatique utilisaient des textes pour des études de fréquences, concordances ou autres. Ces centres avaient tous des conventions particulières pour la mise sur bande. En particulier la longueur des data était fixée soit à un certain nombre de mots-machine, soit à un enregistrement physique. Le format de catalogues, s'il est accepté par ces centres va donc permettre une généralisation.

B - RESULTATS DE L'ANALYSE MORPHOLOGIQUE DU CORPUS

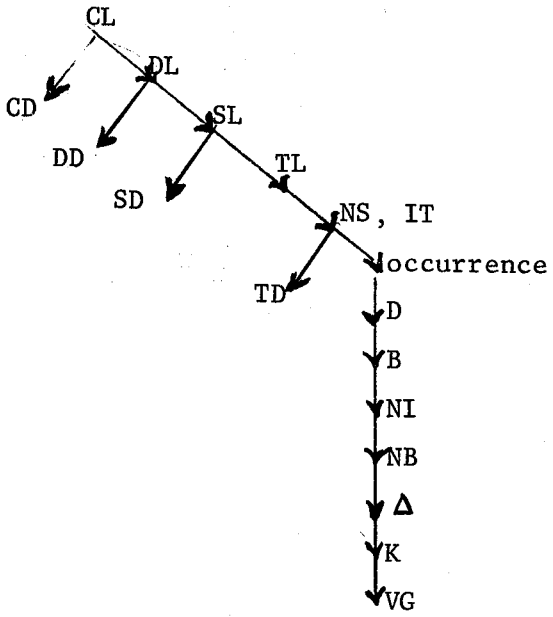
Nous nous souvenons que les trois phases de traduction automat sont les analyses morphologique, syntaxique, sémantique. La première s'occupe la décomposition du mot en sa base et ses affixes,la seconde de la structure d la phrase, la troisième de son sens.

Nous nous intéressons, ici, à l'analyse morphologique d'un corpus donné, destinée à fournir les informations nécessaires au programme syntaxique, puis au programme sémantique.

On appelle occurrence du texte l'information qui se trouve ent deux blancs. A chaque occurrence sont associés :

- un numéro séquentiel NS qui caractérise sa place dans le texte.
- des informations typographiques IT précisant s'il s'agit de majuscules ou minuscules.
- les divers découpages D possibles de l'occurrence.
- les bases B obtenues dans chaque découpage.
- un numéro d'identification NI caractérisant la classe morphologique et l'unité lexicale.
- un numéro de base NB
- un numéro de dérivation Δ permettant de savoir comment chaque base se dérive.
- la catégorie syntaxique K.
- les variables grammaticales VG.

Finalement la carte sera :



Mis sous cette forme, le corpus sera directement utilisable par le programme d'analyse syntaxique. Il peut évidemment s'avérer indispensable de changer cette structure. On doit donc s'attendre à utiliser pour ce faire, une transformation. Vraisemblablement, dans ce cas, on changera l'ordre des classes de "occurrence" à "VG" et on utilisera une TE1 qui est une transformation réversible

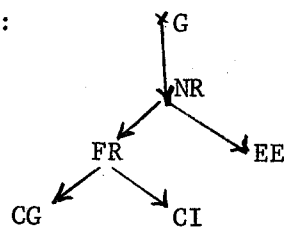
GLOSSAIRES SOUS FORMAT DE CATALOGUE

A moins brève échéance, il est aussi question de mettre certains glossaires G sous format standard.

A un numéro de référence NR on trouvera attachés ses formes en russe FR et ses équivalents anglais EE.

Sous chaque FR on trouvera des codes grammaticaux CG et des codes d'idiomes CI.

La carte sera :



D - LANGAGE DE CATALOGUES

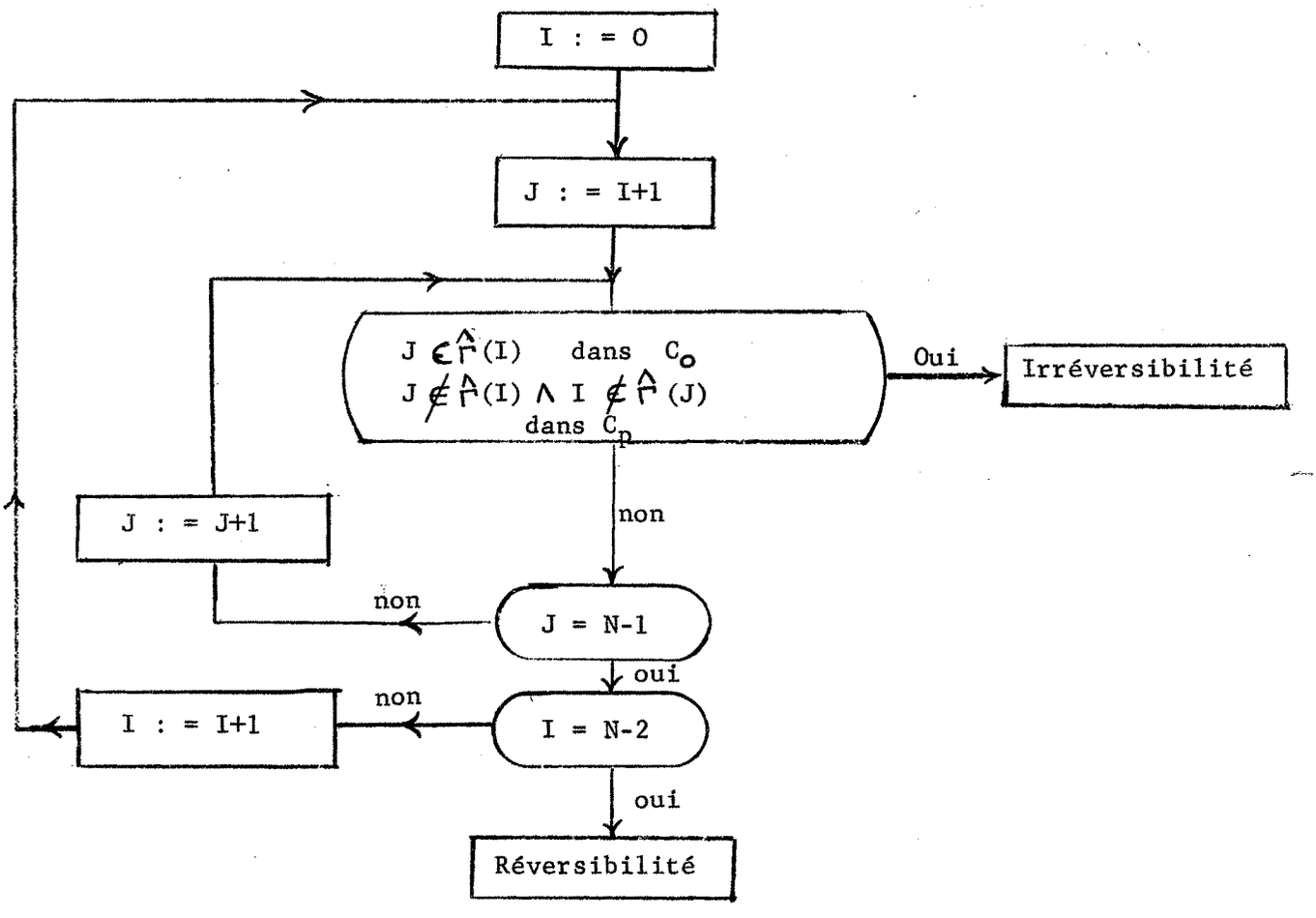
Nous voudrions ici dégager quelques idées sur l'écriture d'un langage de catalogues. Sans passer par le langage machine ou le langage d'assemblage, il serait intéressant pour un utilisateur de pouvoir écrire en quelques instructions simples, un grand nombre sinon toutes opérations s les catalogues.

Nous allons nous intéresser, dans cette optique, aux opérations décrites dans cette thèse, à savoir les transformations et modifications.

1) Transformations

Le théorème 1 nous permet de décider si une transformation es réversible.

Si on a N classes, de 0 à N-1 l'organigramme de décision est le suivant



Si une transformation est irréversible nous avons vu au chapitre II comment la décomposer en une succession de transformations TE1, TE2, TE3, TE4. Nous exposerons seulement l'algorithme de décomposition dans le cas réversible.

Le programme général détermine quelle est la succession des transformations TE1, TE2, TE3 à effectuer et l'utilisateur fait appel aux sous-programmes requis pour la prochaine opération.

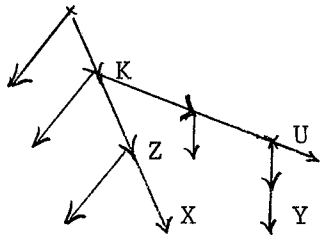
Nous avons créé dans cette optique une procédure ALGØL : PRØTRAELEM permettant ce travail.

Au début, on donne C_o et C_p, à PRØTRAELEM qui nous les rend, accompagnés de C_a et nous décrit la transformation élémentaire de C_o à C_a. On recommence ce processus jusqu'à ce que C_a soit la carte C_p.

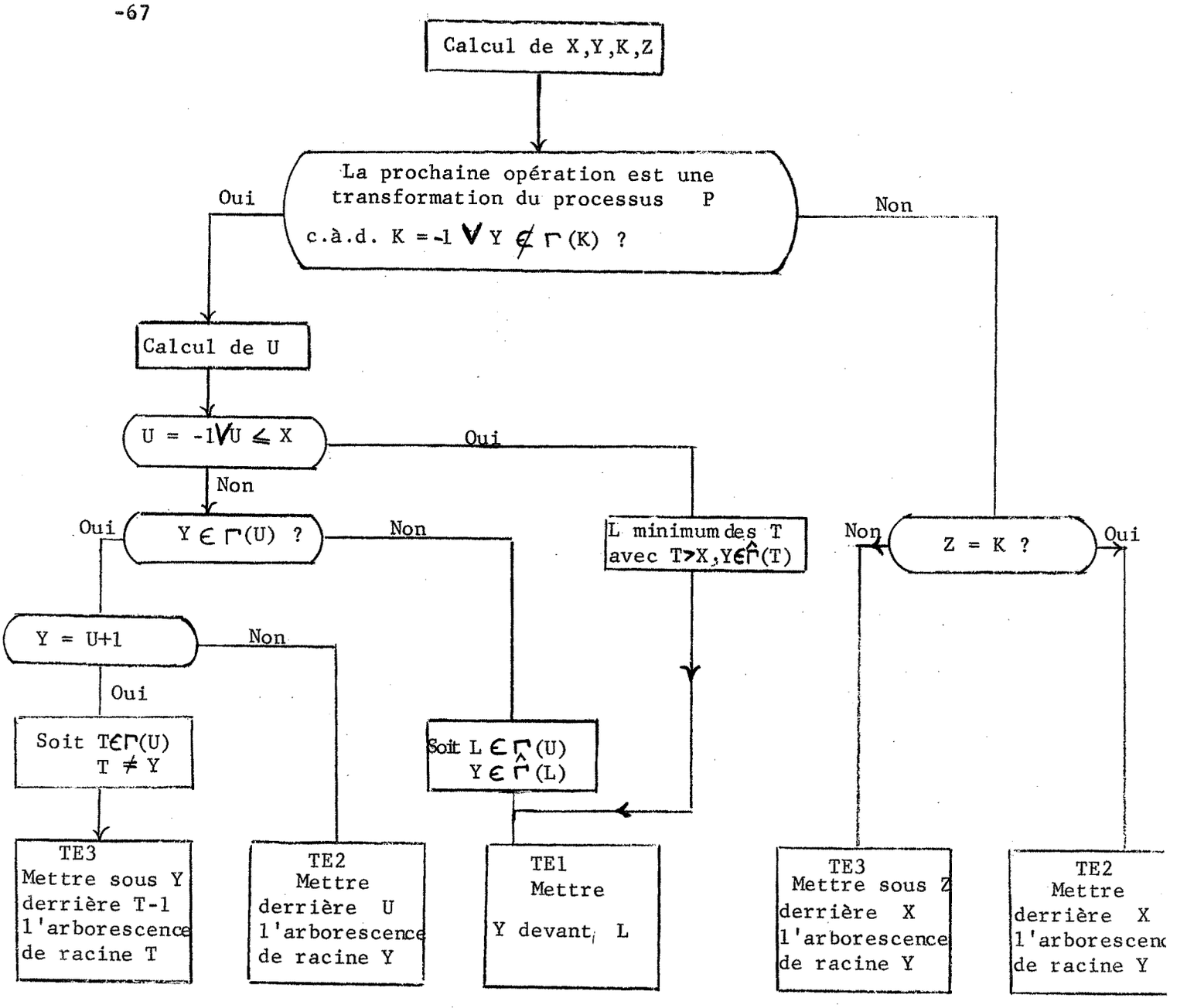
Analyse de la procédure PRØTRAELEM

Travaillant en ALGØL, nous donnerons aux classes des noms numériques plutôt qu'alphabétiques. C_o sera noté CO, C_p : CP et C_a : CA. Les informations de départ sont les cartes CD et ND, tableaux de 0 à N-1, donnant le nom et le numéro de niveau d'une carte, ainsi que CP et NP pour la carte finale à obtenir. La première transformation élémentaire est donnée par CA et NA.

La procédure suit pas à pas le processus décrit dans le théorème 1. X est la dernière classe bien rangée, Y la prochaine classe, à ranger sous Z derrière X. En remontant de Y vers K déjà rangé, U est la première classe à plusieurs descendants. Il est évidemment possible que K, U, Z n'existent pas, auquel cas on leur attribuera la valeur -1.

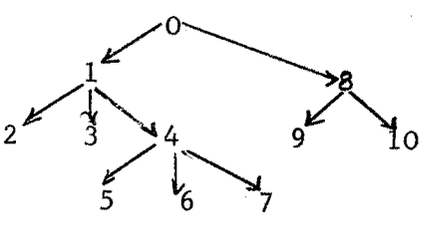


Nous donnons ici l'organigramme de PRØTRAELEM.

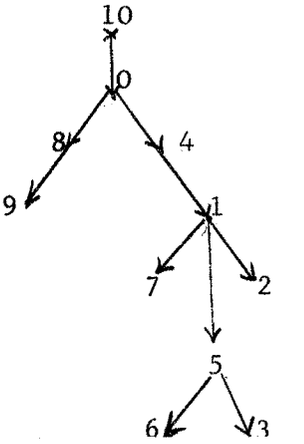


Ce programme a été essayé sur la transformation T_{op} avec

C_o :



C_p :



PRØTRAELEM détermine une succession de 13 transformations élémentaires pour aller de C_o à C_p.

Pour des petits catalogues en particulier, il est intéressant qu'un programme gère une transformation d'un bout à l'autre. Pour des catalogues tenant sur plusieurs bandes, l'utilisateur devra lui-même faire directement appel à des sous-programmes par des instructions telles que :

DETACHER L ARBORESCENCE DE RACINE Y PØUR LA METTRE SOUS Z DERRIERE X.
CHANGER L ORDRE DES CLASSES A, B, C EN C, B, A.
...
SUPPRIMER LA CLASSE D.

Nous donnons en appendice la procédure PRØTRAELEM.

2) Modifications

a) Pour désigner un datum, trois renseignements peuvent paraître intéressants, sa classe C, sa valeur V, son numéro d'occurrence N parmi les data de mêmes classes ayant le même gouvernant.

On désignera un datum par (C,V N)

Un des renseignements peut être omis.

On peut avoir (C,,N),(C,V,).... On définira ainsi un datum et son contexte pour modifier un catalogue. Dans l'exemple proposé page les instructions deviennent :

AJØUTER (2, c_2,) SOUS (o, a_1) .
AJØUTER (o, a_4,) (1, b_3,) (2, c_6,) .
SUPPRIMER (1, b_2,) SØUS (o, a_1,) .
DELAISSER (o, a_2,) .
REMPLACER (o, a_3,) PAR (o, a_6,).

b) Plus généralement on appellera des catalogues de modifications d'un catalogue donné.

En effet, on peut s'attendre à ce que toutes les modifications ne tiennent pas en mémoire centrale. On aura des catalogues d'addition, suppression, deletion, remplacement. Pour ce dernier cas, on admettra que la succession de deux data de même classe représentent, le premier, le datum à remplacer, le second, le datum par lequel on le remplace.

On pourrait trouver :

AJOUTER LE CATALOGUE A AU CATALOGUE B.

SUPPRIMER LE CATALOGUE A DU CATALOGUE B.

etc ...

c) On peut désirer trouver un catalogue d'une carte donnée dans un catalogue de catalogue, ou seulement y trouver une certaine classe.

TROUVER LA CARTE C DANS LE CATALOGUE A.

TROUVER LA CLASSE C DANS LE CATALOGUE A.

OU

SAUTER UN CATALOGUE etc...

3) Concordances

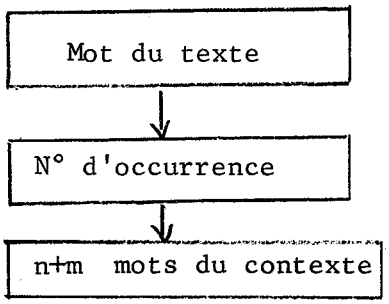
Dans les études de sémantique en traduction automatique et plus généralement de mécano-linguistique, il est intéressant de souligner quels sont les mots les plus habituels dans le contexte d'un mot donné.

Prenons le cas du mot français "glace". Sa traduction sera évidemment différente si on rencontre dans le contexte les mots "banquise" ou "armoire".

Si toutefois un linguiste cherche à définir quels sont les mots qu'on rencontre le plus fréquemment avec un mot donné, son étude sera forcément subjective.

Il semble plus intéressant de faire faire ce travail par un programme à partir d'un choix étendu de textes.

A chaque mot on peut associer les n mots qui le précédent et les m mots qui le suivent.



Ce programme nécessitera un tri, car on rencontre un mot donné à plusieurs endroits du texte.

4) Idée de syntaxe éventuelle d'un langage de catalogues

En nous inspirant d'Algol, nous pouvons garder la structure de blocs avec déclarations et instructions. Tout catalogue sur lequel on doit faire des opérations doit être déclaré puis ouvert de manière à ce que la carte soit lue si c'est un catalogue de lecture, donnée puis écrite si c'est un catalogue d'écriture. On fermera les catalogues avant la fin du programme.

Exemple

```
début
catalogue  NØM1 , NØM2 ;
Carte C1 , C2 ;

OUVRIR LE CATALØGUE DE LECTURE (NØM1 , C1) ;
C2 [0] : =  MOT, NIVEAU 1 ; C2 [1] : = OCCURRENCE, NIVEAU 2 ;
C2 [2] : =  CØNTEXTE,NIVEAU 3 ;
OUVRIR LE CATALØGUE D ECRITURE  (NØM2 , C2) ;
CØNCØRDANCES NØM1  SUR NØM2 ;
FERMER LE CATALØGUE DE LECTURE NØM1;
FERMER LE CATALØGUE D ECRITURE NØM2
fin ;
```

CONCLUSION

Les catalogues ont été décrits à l'aide des notions d'arbres et de listes. Ils paraissent devoir s'adapter à toutes opérations de traduction automatique et de mécano-linguistique et même, vraisemblablement, de gestion.

L'étude algébrique permet de concevoir et d'analyser les transformations. Les programmes écrits réalisent les principales opérations concernant un catalogue.

Cette étude est susceptible de s'élargir aux opérations concernant plusieurs catalogues telle la composition, par exemple.

En outre les problèmes traités ici, se soucient davantage de l'arrangement des data entre eux que de leur contenu. On peut alors envisager une bibliothèque de sous-programmes permettant d'opérer en même temps sur les structures et sur les data.

Il semble, donc, intéressant d'imaginer un langage facilitant le travail des utilisateurs. La souplesse de structure étant d'ores et déjà démontrée, l'avenir des catalogues est lié à la commodité de leur manipulation.

BIBLIOGRAPHIE

(1) Claude BERGE - "Théorie des Graphes et ses applications"
Dunod 1958.

(2) GASTINEL-BOLLIET-LAURENT - "Un nouveau langage scientifique Algol"
Hermann 1964.

(3) Anatol HOLT - "A mathematical and applied investigation of tree structures for computer syntactic analysis"
Université de Pennsylvanie 1963.

(4) M. KAY et T. ZIEHE - "Natural language in computer form"
Rand Corporation - Février 1965.

(5) C. PICARD - "Théorie des questionnaires"
C.N.R.S. - Publication N° 19-3-3/BI - Mai 1963.

(6) B. VAUQUOIS et J. VEYRUNES - "Présentation de l'analyse morphologique du russe".
C.E.T.A.G - Document G.100.C.

(7) Gérard VEILLON - "Introduction à la théorie des catalogues"
C.E.T.A.G - Document G.2100.B - Janvier 1965.

(8) IFFIP/WC N°1 - (IFIP/ICC TERMINØLØGY)

(9) CIØS - "Rand catalog Input-Output System"
Aout 1965 - Rand Corporation (A paraître).

APPENDICE

PROCEDURE PRØTRAELEM

```
Procédure PRØTRAELEM (CD,ND,CP,NP,CA,NA,N,UT,VT,WT,CØDE) ;
entier tableau CD,ND,CP,NP,CA,NA ; entier N,UT,VT,WT,CØDE ;
commentaire CETTE PRØCEDURE UTILISE LA CARTE DE DEPART CD,ND, ØU CD REPRESENTE LES
      NØMS DE CLASSE, ND LES NIVEAUX. CP,NP EST LA CARTE FINALE. CES TABLEAUX VØNT DE
      0 A N-1.
      CØDE = 1 SI C EST UNE TE1 : ØN CHANGE LA BRANCHE DE UT A VT. CØDE = 2 PØUR TE2 :
      ØN MET SØUS UT, DERRIERE WT L ARBØRESCENCE DE RACINE VT. CØDE = 3 SI C EST TE3 :
      ØN MET SØUS UT, DERRIERE WT, L ARBØRESCENCE DE RACINE VT, UT,VT,WT, SØNT DES
      NUMERØS, INDICES DE CD,ND ;

début procédure GAMMA (U,MG,NØMB) ;
      entier tableau MG ; entier U,NØMB ;
      commentaire CETTE PRØCEDURE MET DANS LE TABLEAU MG, DE 1 A NØMB LES ELEMENTS DE
           GAMMA (U) PAR RAPPØRT A CD,ND ;
      début entier I,J ; NØMB :=0 ; J :=ND [U] + 1 ;
            pour I := U+1 pas 1 jusqua N-1 faire
            début Si ND [I] < J alors aller a SØRGA ;
                  Si ND [I] = J alors
                  debut NØMB := NØMB + 1 ; MG [NØMB] :=I
                  fin
            fin ; SØRGA :
      fin GAMMA ;

      booléen procédure ASCENDANT (A,B) ; entier A,B ;
      commentaire ASCENDANT EST VRAI SI A EST ASCENDANT DE B DANS CD,ND ;
      début entier I,NA,NB,NI ;
            NA := ND [A] ; NB := ND [B] ;
            Si B≤A ∨ NA≥NB alors
            début ASCENDANT := faux ; aller a FINAS
            fin ; I := B ; UR : I :=I-1 ; NI :=ND [I] ;
            Si NI ≠ NA alors aller a UR ;
            Si I = A alors ASCENDANT := vrai
```

```
sinon  ASCENDANT := faux ; FINAS :
fin  ASCENDANT ;

procédure NØUCAR1 (L,Y) ; entier L,Y ;
commentaire CREE CA,NA, A PARTIR DE CD,ND DANS UNE TE1 OU Y EST PLACE DEVANT
   L,Y ET L SØNT DES NUMERØS PAR RAPPØRT  A CD ;
début entier I ;
     pour I := 0 pas 1 jusqua  N-1 faire
     NA [I] :=ND [I];
     pour I := 0 pas 1 jusqua  L-1 faire
     CA [I] := CD [I] ;
     CA [L] := CD [Y] ;
     pour I :=1 pas 1 jusqua  Y-L faire
     CA [L+I] := CD [L+I-1] ;
     pour I := Y+1 pas 1 jusqua N-1 faire
     CA [I] := CD [I] ;
fin  NØUCAR1 ;

procedure  NØUCAR2 (U,Y) ; entier Y,U ;
commentaire  CETTE PRØCEDURE CREE CA,NA AVEC CD,ND PØUR UNE TE2 OU L ARBORESCEI
   DE RACINE Y, DE ALPHA ELEMENTS, EST PLACEE DERRIERE  U ;
début entier  I, ALPHA,V ;
     pour I := 0 pas 1 jusqua  U  faire
     début  CA [I] :=  CD [I] ;  NA [I] := ND [I]
     fin ; ALPHA := 1 ;
     pour  I := Y+1  pas 1 jusqua  N-1 faire
     Si  ND [I]⩽ND [Y] alors  aller a  FIAL
     sinon  ALPHA := ALPHA+1 ;
     FIAL : pour I := 1 pas 1 jusqua  ALPHA faire
     début  CA [U+I]  := CD [Y+I-1] ;
            NA [U+1]  := ND [Y+I-1] ;
     fin ; V := U+ALPHA ;
     pour I := 1 pas 1 jusqua Y-U-1 faire
     début CA [V+I]  := CD [U+I]  ;
           NA [V+I]  := ND  [U+I]
```

```
        fin ; pour I := Y+ALPHA pas 1 jusqua N-1 faire
        début CA [I] :=  CD [I] ; NA [I] := ND [I]
        fin
fin NØUCAR2 ;

procedure NØUCAR3 (A3,B3,C3) ; entier A3,B3,C3 ;
commentaire  CREE CA,NA AVEC CD,ND PØUR UNE TE3 ØU L ARBØRESCENCE DE RACINE B3, DE
   ALPHA ELEMENTS EST PLACEE SØUS A3 DERRIERE C3 ;
début entier  I, ALPHA, V ;
        ALPHA :=1 ; pour I := B3+1 pas 1 jusqua N-1 faire
        Si ND [I] ⩽ ND [B3] alors aller a FIA
        sinon  ALPHA := ALPHA+1 ; FIA : V := C3+ALPHA ;
        pour I := 0 pas 1 jusqua C3 faire
        début  CA [I] :=  CD [I] ;  NA [I] :=  ND [I]
        fin ; pour I := 1 pas 1 jusqua ALPHA faire
        début  CA [C3+I] :=  CD [B3+I-1] ;
               NA [C3+I] :=  ND [B3+I-1] +ND [A3] -ND [B3] +1
        fin ; pour I := 1 pas 1 jusqua B3-C3-1 faire
        début  CA [V+I] := CD [C3+I] ;
               NA [V+I] := ND [C3+I]
        fin ;
        pour I := B3+ALPHA pas 1 jusqua N-1 faire
        début  CA [I] := CD [I]; NA [I] := ND [I]
        fin
fin NØUCAR3 ;

entier I,J,Z,K,U,X,Y,T,D,L ;
entier tableau  MG [0 : N-1] ;
commentaire X := MAX,I,CD [I] = CP [I],Y  SUIT X DANS CP, Y ATTACHE A Z DANS CP.
   K⩽X ET Y APPARTIENT A L ARBØRESCENCE DE RACINE K DANS CD. U PREMIERE CLASSE
   TRØUVEE EN REMØNTANT CD DEPUIS Y TELLE QUE GAMMA(U) AIT PLUSIEURS ELEMENTS ;
pour I :=0 pas 1 jusqua N-1 faire
Si  CD [I] ≠ CP [I] ⋁ ND [I] ≠ NP [I] alors
début  X := I-1 ;  aller a TRØUVX
fin ; TRØUVX : I := CP [X+1] ;
pour J := X+1 pas 1 jusqua N-1 faire
Si  CD [J] = I alors
```

```
début  Y := J ; aller a TRØUVY
fin ; TRØUVY : J:= NP [X+1]  ; Z := X+1 ;
REZ : Z := Z-1 ; Si Z =-1  alors aller a  TRØUVZ ;
Si NP [Z] ≠ J-1 alors aller a REZ ;
TRØUVZ : K := Y ; REK : K := K-1 ; Si K =-1  alors aller a PRP ;
Si  K>X ∨ ¬ ASCENDANT (K,Y)  alors aller a REK ;
GAMMA (K,MG I) ;
pour J := 1 pas 1 jusqua I faire
Si MG [J] = Y  alors aller a NØN PRP ;
aller a PRP ;
NØNPRP : commentaire  CE N EST PAS UNE TRANSFØRMATIØN  DU PRØCESSUS P ;
Si K=Z alors aller a  CAS1 ;
CAS5 : CØDE :=3 ; UT :=Z ; VT :=Y ; WT :=X ;
NØUCAR3 (UT,VT,WT) ; aller a FPRØTRA ;
CAS1 : CØDE :=2 ; UT :=Z ; VT :=Y ; WT :=X ;
NØUCAR2 (X,Y) ; aller a FPRØTRA ;
PRP : commentaire C EST UNE TRANSFØRMATIØN DU PRØCESSUS P ; U :=Y ; I :=ND [Y] -1
RECU : U :=U-1 ; Si U<0  alors
début L:= X+1 ; aller a PØT1
fin ; Si  ND [U] = I  alors
début  I :=I-1 ; GAMMA (U,MG,J) ;
      Si J=1 alors aller a RECU sinon
      aller a  TRØUVU
fin ; aller a RECU
TRØUVU : Si U≤X  alors aller a  CAS2 ;
pour I :=1 pas 1 jusqua  J faire
Si MG [I] =Y  alors aller a  YSUITILU ;
aller a  CAS2 ;
YSUITILU : Si Y ≠ U+1  alors aller a  CAS3 ;
CAS4 :  T := MG [2] ; CØDE :=3 ;
UT :=Y ; VT :=T ; WT :=T-1 ;
```

```
        NØUCAR3 (UT,VT,WT) ; aller a FPRØTRA ;
        CAS3 : CØDE :=2 ; UT :=U ; VT :=Y ; WT :=U ;
        NØUCAR2 (UT,VT) ; aller a FPRØTRA ;
        CAS2 : GAMMA (U,MG,J) ;
        pour I :=1 pas 1 jusqua J faire
        début L := MG [I] ; Si ASCENDANT (L,Y) alors
                aller a PØT1
        fin ;
        PØT1 : CØDE :=1; UT :=L ; VT :=Y ; NØUCAR1 (L,Y) ;
        FPRØTRA :
fin DE PRØTRAELEM ;
```

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