ANNALES DE L'I. H. P., SECTION B

F. BENZÉCRI

Linguistique mathématique : l'algèbre des constituants non-connexes

Annales de l'I. H. P., section B, tome 3, n° 1 (1967), p. 1-34

<http://www.numdam.org/item?id=AIHPB_1967__3_1_1_0>

Ann. Inst. Henri Poincaré,
Vol. III, n° 1, 1967, p. 134.

Section B :
Calcul des Probabilités et Statistique.

Linguistique mathématique : l'algèbre des constituants non-connexes

par

F. BENZÉCRI

Dans son cours de linguistique mathématique, J. P. Benzécri a défini une catégorie dont les morphismes sont des opérations d'imbrication portant sur des listes de mots non connexes.

On donne ici un langage de programmation qui permet de décrire ces opérations auxquelles on a conservé le nom de morphismes.

H. Vandenburgh a écrit un programme de macroinstructions Fortran II qui permet d'exécuter effectivement sur machine une suite d'opérations formulées dans notre langage.

1. MOTS ET MONOÏDES

Étant donné un ensemble Λ (appelé alphabet) d'éléments (appelés lettres) nous appellerons mot sur Λ toute suite finie d'éléments (distincts ou non) de Λ :

$$\lambda_1 \lambda_2 \ldots \lambda_i \ldots \lambda_n$$

où $(1, 2, \ldots, i, \ldots, n)$ est la suite des n premiers entiers et, pour tout i compris entre 1 et n, λ_i est une lettre de Λ :

$$\forall i, \quad \lambda_i \in \Lambda.$$

Le nombre, n, de termes de la suite est appelé longueur du mot. Il n'y a qu'un seul mot de zéro lettre : le mot vide que nous noterons, si c'est nécessaire, par $\emptyset$. Il y a autant de mots d'une seule lettre que de lettres et, du fait des notations adoptées, un mot d'une lettre et sa lettre s'écrivent pareil.

EXEMPLES

1° Λ est l'alphabet des minuscules latines.

etre

est un mot de quatre lettres (ou de longueur 4).

2° Λ est l'ensemble des éléments suivants :

— les mots français écrits;

— le « blanc » qui sépare deux mots dans un texte et que nous noterons # pour le désigner plus commodément;

— les signes de ponctuation;

le#soleil#resplendit.

est un mot de six lettres (ou de longueur 6).

On peut multiplier les exemples en prenant pour Λ l'ensemble des syllabes ou des phonèmes d'une langue, les idéogrammes de langues idéogrammatiques, etc.

En vue du traitement sur machine, on prend ici pour Λ l'ensemble des éléments suivants, figurant au clavier de la machine :

— les capitales latines
— le blanc : #
— les signes de ponctuation . et ,
— les chiffres

à l'exclusion des signes : = — @ $ / + * ( ) réservés à d'autres usages.

Sur l'ensemble $L(\Lambda)$ (ou simplement L) de tous les mots sur Λ, on définit l'opération « produit de juxtaposition » qui, au couple ordonné de deux mots $(\lambda, \mu) \in L \times L$, où

$$\lambda = \lambda_1 \ldots \lambda_n; \qquad \mu = \mu_1 \ldots \mu_p$$

associe le mot, noté $(\lambda\mu) = \xi$,

$$\lambda_1 \ldots \lambda_n \mu_1 \ldots \mu_p = \xi_1 \ldots \xi_{n+p}$$

où

$$\xi_i = \lambda_i \quad \text{pour} \quad 1 \leqslant i \leqslant n \quad \text{et} \quad \xi_i = \mu_{i-n} \quad \text{pour } n+1 \leqslant i \leqslant n+p.$$

Cette loi est associative :

Soit $\lambda = \lambda_1 \ldots \lambda_n$, $\mu = \mu_1 \ldots \mu_p$, $\nu = \nu_1 \ldots \nu_r$ trois mots sur Λ ; l'on a :

$$((\lambda\mu)\nu) = (\lambda(\mu\nu)) = \lambda_1 \ldots \lambda_n \mu_1 \ldots \mu_p \nu_1 \ldots \nu_r$$

$L(\Lambda)$, muni du produit de juxtaposition, n'est autre que le monoïde libre associé à Λ.

REMARQUES

1° Le produit de juxtaposition n'est pas une loi commutative :

$$\lambda_1 \ldots \lambda_n \mu_1 \ldots \mu_p \neq \mu_1 \ldots \mu_p \lambda_1 \ldots \lambda_n$$

2° Le mot vide est élément neutre pour cette loi :

$$\lambda_1 \ldots \lambda_n \emptyset = \emptyset \lambda_1 \ldots \lambda_n = \lambda_1 \ldots \lambda_n$$

2. MOTS NON CONNEXES ET LISTES DE MOTS NON CONNEXES

2.1. Mots non connexes.

Mettons en parallèle la phrase latine :

nec adulatoribus latus præbes,

et sa traduction française :

et tu ne prêtes pas le flanc aux flatteurs.

Au mot latin :

nec

correspondent, en français, les trois mots non consécutifs :

et ... ne ... pas;

à :

præbes

correspondent :

tu ... prêtes.

Ainsi, apparaissent comme des unités de la langue, au même niveau que les mots, des suites de mots non nécessairement consécutifs : les « mots non connexes ».

Formellement, étant donné un alphabet Λ (cf. 1°), nous appellerons mot non connexe à n composantes (n entier positif non nul) sur Λ toute suite de n mots sur Λ (les n composantes).

L'ensemble de tous les mots non connexes sur Λ n'est autre que l'ensemble $L(L(\Lambda))$ de toutes les suites finies d'éléments de $L(\Lambda)$. On identifiera tout mot non connexe à une seule composante au mot sur Λ qui le constitue : $L(\Lambda)$ est alors identifié à une partie de $L(L(\Lambda))$.

Dans un mot non connexe, les composantes seront séparées par le signe :

—

par exemple :

ATA—BRIQUE
RI—DO—

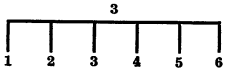
sont des mots non connexes, le premier à 2 composantes, le second à 3 composantes dont la troisième est le mot vide de L(Λ).

Nous ne considérerons pas de mot non connexe à 0 composante. Le mot vide de L(Λ) est identifié à un mot non connexe à une composante. Pour chaque valeur entière non nulle de n, il existe un mot non connexe à n composantes toutes vides; par exemple

— — — — —

est un mot non connexe à 6 composantes vides.

Dans les formules, de même qu'en algèbre on désigne une variable par x, (et éventuellement si plusieurs variables interviennent, par x_1, x_2, ...), un mot non connexe à n composantes sera schématisé par un peigne, ou peigne-schéma, (éventuellement affecté d'un numéro) à n dents ordonnées de gauche à droite (et éventuellement numérotées de 1 à n) en sorte que la première dent représente la première composante, la seconde dent la seconde composante, etc. :



2.2. Listes de mots non connexes.

Nous appellerons liste tout élément de L(L(L(Λ))) i. e. toute suite finie de mots non connexes; ces mots seront séparés par le signe :

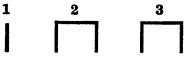
/

par exemple,

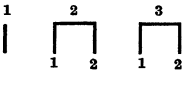
RA/TA—PLAN/PLAN—PLAN

est une liste de trois mots non connexes dont le premier a une composante, les deux autres, deux; on la schématisera par :

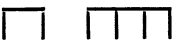
ou, au besoin, par :



ou encore par :

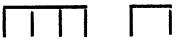


Inversement, la donnée d'un tel schéma ne définit évidemment pas une liste mais un *type* de liste : par exemple, le schéma :



peut représenter une liste quelconque à deux mots non connexes dont le premier a deux composantes, le second quatre. Notons bien que le schéma d'une liste donne l'ordre des peignes. Il faut donc distinguer le type de liste :

et le type :



Cette notion de type de liste permet de classer les listes en sous-ensembles de L(L(L(Λ))) tels que chaque liste appartienne à un et un seul de ces sous-ensembles.

2.3. **En langage de programmation,** pour désigner une liste par un symbole (ensemble de lettres et de chiffres) sans risque de confusion avec le mot de L(Λ) composé des mêmes lettres et chiffres, nous encadrerons ce symbole par deux signes @; par exemple, posons :

$$@\ \text{AB5}\ @ = \text{A—B/C/D—E—F}$$

AB5 est un élément de L(Λ) mais @ AB5 @ désigne une liste d'élément de L(L(L(Λ))), ici, la liste :

$$\text{A—B/C/D—E—F}$$

Un type de liste sera indiqué, non par une suite de peignes (ceux-ci ne figurant pas au clavier de la machine) mais par la liste particulière de ce type dont toutes les composantes sont vides, encadrée par deux signes @; par exemple,

$$@ — / / — — @$$

désigne le type de liste schématisé par

La notation @ ... @ peut en particulier servir à désigner un mot à une ou plusieurs composantes considéré comme liste d'un seul mot.

2.4. **Les notations suivantes** s'interprètent facilement car elles généralisent aux listes représentées par des symboles des opérations dont le sens est clair quand il s'agit des listes elles-mêmes (i. e. d'expressions ne comportant pas de symbole @ ... @).

Soient deux listes :

@ LISTE 1 @, @ LISTE 2 @

on notera :

@ LISTE 1 @/@ LISTE 2 @

la liste obtenue en les juxtaposant (au sens de L(L(L(Λ))), i. e. en les mettant bout à bout, séparées l'une de l'autre par le signe /).

De même, soient deux mots :

@ MOT 1 @, @ MOT 2 @

l'un à p, l'autre à q composantes; on notera :

@ MOT 1 @—@ MOT 2 @

le mot à $p + q$ composantes obtenu en les juxtaposant (cette fois, au sens de L(L(Λ)), i. e. en les mettant bout à bout, séparés l'un de l'autre par le signe —); et soit deux mots à une seule composante :

@ MOT 3 @, @ MOT 4 @

on notera :

@ MOT 3 @ @ MOT 4 @

le mot à une composante obtenu en les juxtaposant (au sens de L(Λ)), i. e. en les mettant bout à bout).

Plus généralement, on pourra désigner des listes en juxtaposant (avec interposition éventuelle d'un — ou d'un /) des expressions de types divers représentant des listes. Par exemple, soit l'expression :

A—/B—C—@ X @/@ Y @ UT;

si l'on pose :

@ X @ = D—E/F/G—H

@ Y @ = T

l'expression s'interprète :

A—/B—C—D—E/F/G—H/TUT

3. EXEMPLES D'OPÉRATIONS SUR LES LISTES : LES MORPHISMES

Sur les mots d'une liste, on peut effectuer des opérations, ou morphismes, dont nous donnerons d'abord des exemples (qui ne soient pas du type juxtaposition comme ceux que l'on vient de considérer).

3.1. **Considérons la phrase française :**

je ne veux pas.

Elle s'écrit, dans l'alphabet Λ :

JE#NE#VEUX#PAS.

et constitue ainsi un mot sur Λ. Ce mot peut être construit à l'aide des mots non connexes suivants :

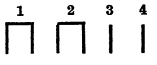
1° JE—VEUX

2° NE—PAS

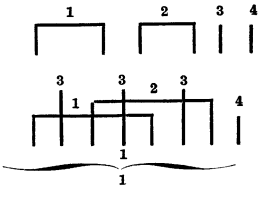
3° # (mot non connexe à une seule composante)

4° .

Ces mots, ainsi ordonnés, constituent une liste schématisée par :



Imbriquons, selon le schéma ci-dessous, les mots non connexes de cette liste, pris chacun une ou plusieurs fois (le # est pris trois fois), en insérant les composantes des uns entre celles des autres, puis juxtaposons (au sens de L(Λ) : cf. 1) les diverses composantes dans l'ordre où l'imbrication précédente vient de les placer.



La première ligne du schéma représente la liste d'où nous partons (nous l'appellerons source du morphisme).

La deuxième ligne indique comment nous avons imbriqué puis juxtaposé les diverses dents de la source (nous l'appellerons graphe du morphisme).

La troisième ligne représente le mot non connexe (ici à une seule composante) obtenu (nous l'appellerons but du morphisme).

L'ensemble du schéma à trois lignes représente le morphisme.

Il est clair que, puisque la première ligne du schéma représente un type de liste, le morphisme peut opérer sur toute liste de ce type. Par exemple, de la liste :

A—B/C—D/E/F

il fait :

AECEBEDF

Ce morphisme conserve, dans chaque mot non connexe, l'ordre de ses composantes (JE précède VEUX, etc.). C'est une condition que l'on peut, ou non, imposer aux morphismes. Si on l'impose, on dira que l'on interdit de croiser les dents des peignes (Nous traitons plus bas un exemple de morphisme qui n'obéit pas à cette condition).

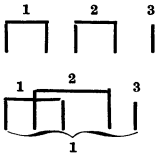
Avant de passer au second exemple, notons que nous aurions pu, pour construire la phrase :

je ne veux pas

partir de la liste :

JE#—VEUX/NE#—#PAS/.

et effectuer le morphisme ci-dessous :



Si l'on veut être sûr que dans toutes les phrases que l'on construira les mots consécutifs seront toujours séparés par un blanc et un seul, l'adjonction de blancs au commencement ou à la fin des mots pose des problèmes parfois difficiles à résoudre.

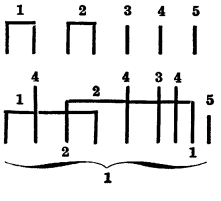
3.2. Considérons maintenant la phrase allemande :

Du räumst dieses ein.

Elle peut être construite à partir de la liste

DU—ST/EIN—RÄUM/DIESES/#/.

sur laquelle opère le morphisme :



Ici, le morphisme permute les dents du peigne 2 de la source.

4. DÉFINITION DES MORPHISMES

4.1. **D'une façon générale,** un morphisme sera défini par deux types de liste de mots non connexes (la source et le but) et un graphe indiquant comment on construit, par le morphisme, une liste de type le but.

Source, but et graphe doivent satisfaire aux conditions de comptabilité données ci-dessous. La source et le but étant schématisés par une suite de peignes comme il a été dit plus haut, le graphe est une suite de dents réparties en peignes, chaque dent du graphe appartenant à un peigne et un seul, et mises en correspondance avec les dents de la source d'une part, celles du but d'autre part, de telle sorte que soient remplies les conditions :

1° A toute dent du graphe correspond une dent unique de la source et une dent unique du but.

2° Aux dents d'un peigne du graphe correspondent, dans la source, autant de dents distinctes constituant un peigne. Au besoin, on indiquera cette correspondance en numérotant, dans la source, les peignes et dans chaque peigne, les dents; peignes et dents du graphe devront porter les numéros de leurs correspondants dans la source.

Si l'on interdit de croiser les dents d'un peigne (cf. *supra*, n° 31) la condition 2° doit être complétée par :

2°′ L'ordre des dents d'un peigne du graphe est celui des dents qui leur correspondent dans un peigne de la source (à la p^e dent d'un peigne du graphe correspond la p^e dent du peigne correspondant de la source).

3° Aux dents d'un peigne du graphe correspondent, dans le but, des dents appartenant à un même peigne; à plusieurs dents consécutives du graphe (que l'on réunit par une accolade) peut correspondre une même dent du but, mais l'ordre des dents doit être respecté en ceci : soient deux dents du graphe, la première située à gauche de la seconde; à la première dent correspond une dent du but qui est ou bien confondue avec la dent correspondant à la seconde, ou bien située à sa gauche.

On disposera les unes sous les autres, les dents se correspondant (ce qui est possible grâce aux accolades réunissant les dents du graphe correspondant à une même dent du but; pour plus de clarté, on pourra aussi mettre une accolade sous une seule dent du graphe).

De la condition 3 il résulte qu'appartiennent à un peigne du but les dents de toute suite de peignes du graphe telle que deux peignes consécutifs de cette suite soient imbriqués (i. e. la première dent de chacun de ces deux peignes est à gauche de la dernière dent de l'autre).

4.2. **Un morphisme étant donné par son schéma,** voici comment il transforme une liste @ X @ de type la source en une liste @ Y @ de type le but. Chaque composante de mot de la liste @ Y @ va être calculée comme produit de juxtaposition de composantes de mots de la liste @ X @. Il est commode, pour notre explication, d'identifier les composantes de mots de @ X @ et de mots de @ Y @ avec les dents qui les représentent dans la source et le but du schéma du morphisme. A une dent du graphe (2e ligne du schéma du morphisme) correspond une dent unique de la source, donc une composante unique de @ X @ que nous considérerons comme représentée par cette dent du graphe. Maintenant, on peut dire que chaque dent du but est le produit de juxtaposition des dents du graphe qui se trouvent réunies au-dessus d'elles par une accolade.

Si l'on distingue dans chaque composante de @ Y @ les composantes de @ X @ dont elle est le produit de juxtaposition, chaque lettre de @ Y @ apparaît comme provenant d'une dent bien déterminée de la source en passant par un peigne bien déterminé du graphe.

Nous appellerons origine d'une lettre de @ Y @ la lettre de @ X @ d'où elle provient et chemin de cette lettre de @ Y @ le peigne du graphe par où elle est passée.

Ainsi, dans l'exemple 3.2 donné ci-dessus, la liste :

$$@\ X\ @ = DU—ST/EIN—R\ddot{A}UM/DIESES/\#/.$$

est transformée par le morphisme en la liste :

$$@\ Y\ @ = DU\#R\ddot{A}UMST\#DIESES\#EIN.$$

la lettre U de RÄUMST dans @ Y @ a pour origine le U de RÄUM dans le second peigne de @ X @ et pour chemin le peigne du graphe portant le numéro 2.

4.3. Morphismes particuliers.

Nous définissons ici des morphismes particulièrement simples à partir desquels pourront être construits tous les autres morphismes par les opérations définies au nº 6.

Les projections :

Les plus simples des projections sont les projections élémentaires dont le schéma se compose :

d'une source quelconque;
d'un graphe, formé par un seul peigne de la source;
d'un but identique au graphe.

L'opération d'une projection élémentaire donnée sur une liste de type sa source consiste à prendre un mot déterminé de cette liste.

Plus généralement, une projection aura pour schéma :

une source quelconque;

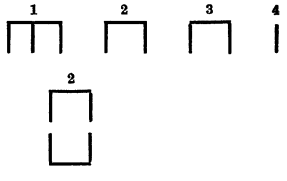
un graphe composé de certains peignes de la source, répétés ou non, placés bout à bout dans un ordre qui peut être différent de celui où ils se trouvent, les uns par rapport aux autres, dans la source;

un but identique au graphe.

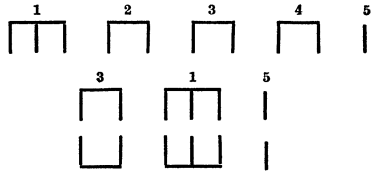
L'opération d'une projection donnée sur une liste de type sa source consiste à prendre certains mots déterminés de cette liste dans un ordre déterminé.

Voici des exemples de projections :

Une projection élémentaire :



Une projection quelconque :



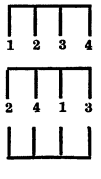
Les permutations (morphismes uninaires i. e. opérant sur des listes d'un seul mot) ont pour schéma :

une source, composée d'un seul peigne;

un graphe, formé de l'unique peigne de la source, pris une seule fois après en avoir, ou non, permuté les dents.

L'opération d'une permutation sur un mot non connexe consiste à changer l'ordre de ses composantes.

Voici un exemple de permutation :



Les liaisons sont aussi des morphismes uninaires; leurs schémas se composent :

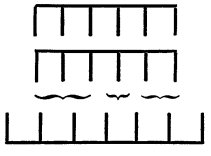
d'une source comprenant un seul peigne;

d'un graphe formé de l'unique peigne de la source pris une seule fois après avoir relié par des accolades certaines dents consécutives;

d'un but formé d'un seul peigne compatible avec le graphe (i. e. sous chaque dent libre, et sous chaque accolade se trouve une dent du but).

L'opération d'une liaison sur un mot non connexe consiste à juxtaposer certaines composantes consécutives pour en faire une seule composante et introduire, ou non, de nouvelles composantes vides.

Voici un exemple de liaison :



Du mot :

A—B—C—D—E—F

ce morphisme fait le mot :

— —ABC— —D—EF—

Les imbrications sont des morphismes binaires : ils opèrent sur des listes de deux mots non connexes; leurs schémas se composent :

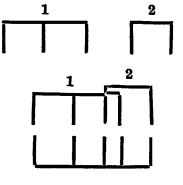
d'une source, comprenant deux peignes seulement;

d'un graphe, formé des deux peignes de la source pris une seule fois en conservant, dans chacun d'eux, l'ordre des dents, et imbriqués, ou non, l'un dans l'autre (cf. fin du n° 4.1);

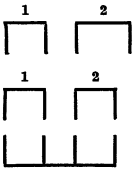
d'un but, composé uniquement des dents du graphe réunies en un seul peigne.

De deux mots non connexes, l'un à p, l'autre à q composantes (p, q, nombres entiers), une imbrication fait un mot non connexe de composantes les $p + q$ composantes des deux mots donnés de telle sorte qu'on retrouve l'un de ces deux mots en supprimant du résultat les composantes de l'autre.

Voici deux exemples d'imbrication :



ou encore :



Ce dernier exemple montre que la juxtaposition de deux mots non connexes est le résultat d'une imbrication particulière portant sur ces deux mots.

4.4. **En langage de programmation,** pour désigner un morphisme par un symbole (ensemble de lettres et de chiffres) nous ferons précéder ce symbole du signe :

$

Par exemple:

$PSI

désignera un morphisme.

Soit une liste @ A @ de type la source de $PSI; la liste transformée de @ A @ par le morphisme $PSI sera notée

$PSI(@ A @).

Le morphisme lui-même sera noté par le type de la liste source suivi, entre deux signes $, d'une suite de triples d'entiers représentant la suite des dents du graphe; les peignes du graphe étant rangés dans l'ordre où se rencontrent leurs premières dents quand on les parcourt de gauche à droite, toute dent du graphe sera représentée par trois entiers séparés entre eux par une virgule :

le rang du peigne du graphe auquel elle appartient,
le rang du peigne correspondant de la source,
le rang, dans le peigne de la source, de la dent correspondant à la dent donnée du graphe.

Le but et la correspondance entre le graphe et le but seront indiqués par les signes :

. — et /

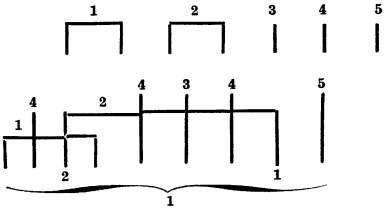
placés entre les triples :

deux triples séparés par la signe . correspondent à une même dent du but,

deux triples séparés par le signe — correspondent à deux dents différentes du but appartenant à un même peigne,

deux triples séparés par le signe / correspondent à deux dents différentes du but appartenant à deux peignes différents.

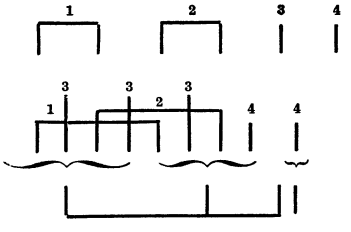
Par exemple le morphisme, que nous allons appeler $PHI, utilisé plus haut pour construire la phrase allemande : « du räumst dieses ein » et schématisé par :



sera ainsi décrit :

$PHI = @—/—///@ $1,1,1.2,4,1.3,2,2.1,1,2.4,4,1.5,3,1.6,4,1.3,2,1.7,5,1$

Autre exemple : Soit $PSI le morphisme schématisé par :



$PSI = @—/—//@ $1,1,1.2,3,1.3,2,1.4,3,1—1,1,2.5,3,1.3,2,2.6,4,1—/7,4,1$

Dans ce dernier exemple, la 3e dent du 1er peigne du but ne correspond à aucune dent du graphe; c'est ce qu'indique, dans la suite des triples d'entiers, la composante vide avant le signe /.

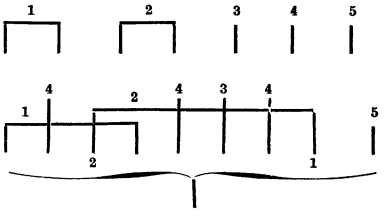
Quand on interdit de croiser les dents des peignes, le troisième nombre du triple représentant une dent du graphe est inutile et l'on peut utiliser une notation, un peu plus simple, à l'aide de couples de nombres. Par exemple,

$PSI = @—/—//@ $1,1.2,3.3,2.4,3—1,1.5,3.3,2.6,4—/7,4$

4.5. **Nous avons donné,** déjà, deux manières de noter les morphismes : l'une schématique et intuitive, l'autre utilisable en programmation. Nous donnons maintenant une troisième notation par un tableau de nom-

bres entiers qui pourra servir à l'étude de la « composition » des morphismes définie plus bas (n° 62).

Reprenons l'exemple que nous avons appelé $PHI :



Numérotons les dents du graphe, de la source et du but de gauche à droite. Ici, l'on a :

pour la source : 1,2,3,4,5,6,7,
pour le graphe : 1,2,3,4,5,6,7,8,9,
pour le but : 1.

Les peignes du graphe sont rangés, comme plus haut, dans l'ordre de leurs premières dents.

Nous allons dresser un tableau qui décrira complètement le morphisme. Il comprendra, de gauche à droite, trois colonnes principales concernant : la source, le graphe, le but.

Les lignes du tableau correspondront, de haut en bas, à la suite des dents du graphe. Décrivons par exemple la troisième ligne du tableau du morphisme $PHI; elle correspond à la 3e dent du graphe et comprend :

dans la colonne médiane, le numéro i de la dent du graphe considérée (ici $i = 3$), le numéro j du peigne auquel elle appartient (ici $j = 3$), son rang ri dans ce peigne (ici $ri = 1$);

dans la colonne de gauche, le numéro εi de la dent de la source correspondant à la dent du graphe considérée (ici $\varepsilon i = 4$), le numéro ej du peigne de la source auquel elle appartient (ici $ej = 2$), son rang $r(\mathrm{A})\ \varepsilon i$ dans ce peigne (ici $r(\mathrm{A})\varepsilon i = 2$);

dans la colonne de droite, le numéro πi de la dent du but correspondant à la dent du graphe considérée (ici $\pi i = 1$), le numéro pj du peigne du but auquel elle appartient (ici $pj = 1$), son rang $r(\mathrm{B})\pi i$ dans ce peigne (ici 1).

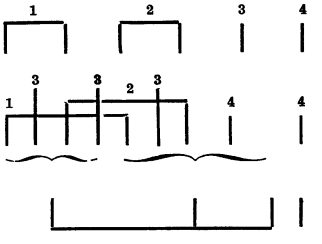
En résumé, voici la troisième ligne du tableau de $PHI :

4 2 2	3 3 1	1 1 1

Donnons maintenant le tableau complet :

εi	ej	$r(A)\varepsilon i$	i	j	ri	πi	pj	$r(B)\pi i$
1	1	1	1	1	1	1	1	1
6	4	1	2	2	1	1	1	1
4	2	2	3	3	1	1	1	1
2	1	2	4	1	2	1	1	1
6	4	1	5	4	1	1	1	1
5	3	1	6	5	1	1	1	1
6	4	1	7	6	1	1	1	1
3	2	1	8	3	2	1	1	1
7	5	1	9	7	1	1	1	1

Donnons encore, comme second exemple le tableau du morphisme $PSI considéré plus haut et dont nous récrivons ici le schéma :



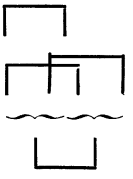
εi	ej	$r(A)\varepsilon i$	i	j	ri	πi	pj	$r(B)\pi i$
1	1	1	1	1	1	1	1	1
5	3	1	2	2	1	1	1	1
3	2	1	3	3	1	1	1	1
5	3	1	4	4	1	1	1	1
2	1	2	5	1	2	2	1	2
5	3	1	6	5	1	2	1	1
4	2	2	7	3	2	2	1	2
6	4	1	8	6	1	2	1	2
6	4	1	9	7	1	4	2	1

4.6. **Remarque :** Au numéro 3.1, nous avons introduit les morphismes comme des opérateurs sur des listes; puis, au numéro 4, nous avons défini formellement les morphismes par leur schéma. Tout morphisme défini par son schéma opère de façon bien déterminée sur toute liste de type sa source pour en faire une liste de type son but. Mais on peut donner un

exemple de deux morphismes ayant même source et même but, dont les schémas sont distincts mais qui coïncident en tant qu'opérateurs (i. e. l'un et l'autre transforment toute liste de type leur source en la même liste).

Soient les deux morphismes :

$MU :



$PI :

Désignons par @ A @—@ B @ une liste quelconque de type ⌜⌝ (@ A @ et @ B @ désignent chacun un mot à une composante sur Λ). L'on a :

$$\$MU(@\ A\ @—@\ B\ @) = \$PI(@\ A\ @—@\ B\ @)$$
$$= @\ A\ @\ @\ A\ @—@\ B\ @\ @\ B\ @.$$

Au numéro suivant nous définirons des mots et listes de mots comportant des signes s'insérant en plusieurs points de la chaîne; deux morphismes de schémas distincts seront distincts en tant qu'opérateurs sur de telles listes (cf. 61).

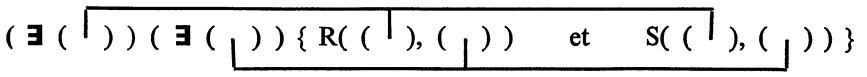
Dans la suite, il nous arrivera d'utiliser, pour désigner un morphisme, les termes de morphisme-schéma ou morphisme-opérateur selon que l'on considérera le morphisme comme schéma à trois lignes (cf. 4.1) ou comme opérateur sur un ensemble de listes.

5. SIGNES A PLUSIEURS INSERTIONS

L'usage de signes à plusieurs insertions n'est pas nouveau. Dans l'introduction de sa *Mathematical Logic* (1957), N. Goodstein, remarque qu'on pourrait se dispenser des variables littérales, en réunissant par un peigne les places où elles s'insèrent dans les formules. Au lieu de :

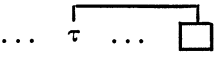
$$(\exists x)\ (\exists y)\ \{\ R(x, y)\quad \text{et}\quad S(x, y)\ \},$$

écrire :

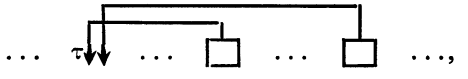


(les parenthèses dont Goodstein encadre les dents nous paraissent d'ailleurs inutiles).

Dans le traité de N. Bourbaki (L. I., Chap. I[er]), on trouve le signe :



qui, s'insérant en deux points d'un assemblage de signes ordinaires, perturbe le caractère monoïdal de la langue de la logique (On rencontre même des τ reliés à un nombre quelconque de carrés pour lesquels, afin d'éviter l'usage d'un vocabulaire infini de signes à plusieurs insertions, on peut proposer une écriture telle que :



l'insertion d'une flèche n'étant possible, dans un assemblage, qu'après une flèche ou un τ).

5.1. **Nous appellerons alphabet gradué** une suite Λ d'ensembles (éventuellement vides) :

$$\Lambda = \{ \Lambda_1, \ldots, \Lambda, \ldots \}$$

où $\boldsymbol{\Lambda}$ (i nombre entier) est l'ensemble des signes à i insertions.

Lorsqu'on voudra n'avoir qu'un nombre fini de signes de base, il faudra imposer à l'alphabet Λ la condition supplémentaire : tous les Λ_i sont vides sauf un nombre fini d'entre eux qui sont finis.

5.1.1. *Un signe à i insertions* (i nombre entier) sera désigné par une capitale latine affectée de l'indice i (on supprimera l'indice 1 pour les signes à une seule insertion de Λ_1) :

$$A_i, B_i, C_i \ldots$$

Les i insertions du signe A_i constituent ce que nous appellerons la matière du signe A_i et seront notées :

$$A_i^1 A_i^2, \ldots, A_i^i ;$$

en sorte qu'à l'alphabet gradué Λ nous associons l'alphabet ordinaire $\widehat{\Lambda}$ constitué par tous les signes effectivement utilisés, à savoir :

les éléments A, B, ... de Λ_1;

les signes A_2^1, A_2^2 ; B_2^1, B_2^2 ... représentant les éléments de Λ_2, etc.;

les signes A_i^1, ..., A_i^i ; B_i^1, ..., B_i^i ... représentant les éléments de Λ_i, etc.

Dans la suite, quand nous parlerons de mots (connexes ou non) et de listes de mots sur $\widehat{\Lambda}$, il s'agira des notions définies en 1 et 2; mais sur Λ ces notions vont être maintenant définies : il faut préciser l'emploi, dans les assemblages, des signes à plusieurs insertions.

5.1.2. *Un mot* (à une composante) sur l'alphabet gradué Λ sera défini par :

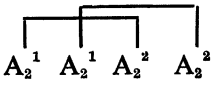
1° un mot sur $\widehat{\Lambda}$, appelé matière du mot sur l'alphabet gradué Λ tel que si une insertion d'un signe de Λ y figure une ou plusieurs fois, toutes les insertions de ce signe y figurent le même nombre de fois;

2° une partition en classes de la suite des lettres de ce mot sur Λ, chaque classe étant la matière d'un signe de Λ; cette partition est indiquée typographiquement par des liens reliant entre eux les constituants d'une même classe (il sera parfois commode de mettre aussi un petit trait vertical, une « dent », au-dessus des signes A, B, etc., de $\Lambda_1 \subset \widehat{\Lambda}$).

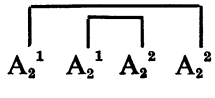
Par exemple, la suite :

$$A_2^1 A_2^1 A_2^2 A_2^2$$

peut servir à définir deux mots distincts sur Λ selon la partition en classes de ses éléments :



ou :



On pourra ou non imposer aux mots sur Λ que les dents des peignes se rencontrent, quand on les parcourt de gauche à droite, dans l'ordre

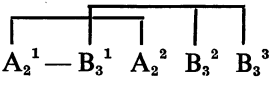
de leurs indices supérieurs (on dira alors, comme pour les morphismes (cf. 4.1), qu'il est interdit de croiser les dents d'un peigne); avec cette condition supplémentaire, $\overline{X_2^1\ X_2^2}$ est un mot sur Λ, mais non $\overline{X_2^2\ X_2^1}$.

5.1.3. *Un mot non connexe à p composantes* (*p*, nombre entier) est défini, de façon analogue, par :

1° un mot non connexe à *p* composantes sur $\widehat{\Lambda}$, appelé matière du mot sur $\widehat{\Lambda}$, tel que si une insertion d'un signe de Λ y figure une ou plusieurs fois, toutes les insertions de ce signe y figurent le même nombre de fois;

2° une partition en classes de la suite de lettres de $\widehat{\Lambda}$ obtenue en juxtaposant les *p* composantes du mot non connexe sur $\widehat{\Lambda}$ donné; chaque classe doit être la matière d'un signe de Λ; cette partition est indiquée typographiquement par des liens (que nous appellerons peignes-liens) reliant entre eux les constituants d'une même classe.

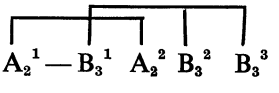
Par exemple,



est un mot non connexe à deux composantes sur Λ; sa matière est le mot non connexe à deux composantes sur $\widehat{\Lambda}$ obtenu en supprimant les peignes-liens :

$$A_2^{\,1} — B_3^{\,1} A_2^{\,2} B_3^{\,2} B_3^{\,3}$$

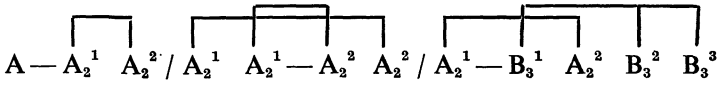
Dans les formules, un mot non connexe sur Λ sera schématisé par le peigne-schéma de sa matière. Ainsi, le mot :



pourra être représenté par le peigne-schéma :

5.1.4. *Une liste de mots non connexes* sur Λ est une suite de mots non connexes sur Λ; deux mots non connexes consécutifs de cette suite sont séparés par le signe : /.

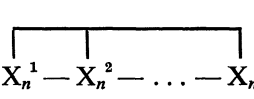
Par exemple :



est une liste de trois mots non connexes sur Λ.

La liste de mots non-connexes sur $\widehat{\Lambda}$ obtenue en supprimant les peignes-liens d'une liste de mots non connexes sur Λ est appelée matière de la liste de mots sur Λ considérée; le type d'une liste de mots sur Λ n'est autre que le type de sa matière.

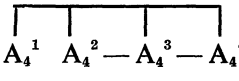
5.1.5. *De même qu'à toute lettre* d'un alphabet ordinaire est associé le mot d'une lettre qu'elle constitue, noté comme cette lettre, de même ici, à tout signe X_n élément de Λ_n (n, nombre entier) est associé le mot à n composantes :



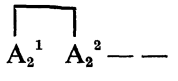
notée aussi X_n (ce mot a pour matière le mot à n composantes sur $\widehat{\Lambda}$:

$$X_n^{\,1} — X_n^{\,2} — \ldots — X_n^{\,n}$$

et la partition de la suite de lettres de $\widehat{\Lambda}$: $X_n^{\,1} X_n^{\,2} \ldots X_n^{\,n}$ ne comprend qu'une seule classe). Tout mot ainsi associé à un signe de Λ est appelé mot-signe. Il est à noter qu'un mot dont la matière provient d'un seul signe peut n'être pas un mot-signe : par exemple :

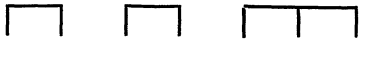


ou

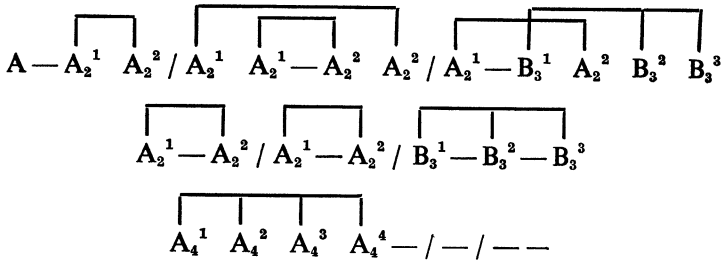


ne sont pas des mots-signes.

5.1.6. *On se gardera de confondre* les peignes-liens qui interviennent dans l'écriture d'un mot non connexe sur Λ ou d'une liste de tels mots et les peignes-schémas du schéma de ce mot ou de cette liste. Par exemple, le schéma :



peut représenter aussi bien l'une quelconque de ces trois listes :



alors que les peignes-liens diffèrent d'une liste à l'autre. Il se trouve que, pour la seconde liste, peignes-liens et peignes-schémas coïncident : c'est que cette liste est formée de mots-signes.

5.1.7. *En langage de programmation,* on ne peut utiliser ni indices, ni liens; nous utiliserons toujours @ ... @ (cf. n° 2.3) pour désigner une liste, un mot (considéré comme liste d'un seul mot), ou même un signe (considéré comme mot-signe) par un symbole, e. g. @ X @ ou @ SIG 6 @, etc.

Dans les alphabets gradués que nous utiliserons, les signes à une seule insertion seront les capitales latines et les chiffres; les signes à plusieurs insertions seront toujours représentés par un symbole entre deux @.

Nous verrons au n° 6.1 que les mots ou listes de mots où entrent des signes à plusieurs insertions peuvent toujours être désignés comme résultats d'opérations de morphismes (cf. 5.2) sur des listes de mots-signes.

5.2. **Les morphismes définis aux numéros 3 et 4** peuvent opérer sur les listes que nous avons définies au n° 5.1.4.

Soit un morphisme $RO donné par son schéma; @ X @ une liste de mots sur Λ de type la source. Nous allons définir la liste de mots sur Λ, @ Y @, transformée de @ X @ par $RO.

Notons @ $\widehat{X}$ @ la matière de @ X @, @ $\widehat{Y}$ @ la liste de mots sur $\widehat{\Lambda}$ transformée de @ $\widehat{X}$ @ par $RO (cf. 4.2).

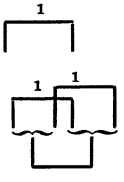
@ $\widehat{Y}$ @ sera la matière de @ Y @. Il reste à définir la partition de l'ensemble des lettres de @ $\widehat{Y}$ @ définissant @ Y @.

A toute lettre de @ $\widehat{Y}$ @ sont associés une lettre bien déterminée de @ $\widehat{X}$ @, son origine, et un peigne bien déterminé du graphe de $RO, son chemin. Deux lettres de @ $\widehat{Y}$ @ appartiendront à une même classe (et seront donc reliées par un peigne-lien) si leurs origines respectives sont reliées par un peigne-lien dans @ X @ et si leurs chemins coïncident.

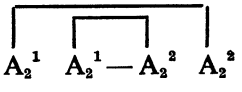
Vérifions que l'on définit bien ainsi une liste de mots non connexes sur l'alphabet gradué Λ : il faut montrer que toute classe sur @ $\hat{Y}$ @ comprend exactement la matière d'un signe. Les lettres d'une classe de @ $\hat{Y}$ @ sont toutes passées par un même chemin (peigne du graphe de $RO). Or, les lettres passées par un même chemin sont exactement celles de @ $\hat{X}$ @ dont ce peigne du graphe porte le numéro. Ainsi une classe de lettres de @ $\hat{Y}$ @ correspond biunivoquement à une classe de lettres de @ $\hat{X}$ @ définissant un signe de @ X @.

Si l'on impose, dans la formation des mots sur l'alphabet gradué Λ, la condition restrictive de ne pas croiser les dents d'un peigne (i. e. de ne pas permuter les insertions d'un même signe) on ne peut faire opérer sur une liste que des morphismes satisfaisant à la condition 2$^{o\prime}$ du n^{o} 4.1. Autrement, dans une classe de lettres de @ $\hat{Y}$ @ les insertions d'un même signe pourraient se trouver permutées.

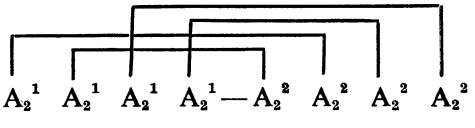
Par exemple, le morphisme $MU du n^{o} 4.6 de schéma :



opère sur la liste de mots non connexes sur Λ :

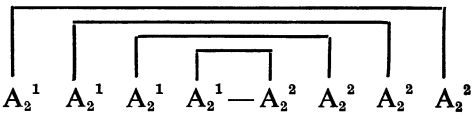


pour en faire la liste :



Le morphisme $RO (cf. n^{o} 4.5) de schéma :

transforme cette même liste en la liste :



6. OPÉRATIONS SUR LES MORPHISMES ; ÉCHELLES

6.1. Caractérisation d'un morphisme par son action sur les listes.

Au nº 4.6, nous avons remarqué que deux morphismes distincts (i. e. dont les schémas sont distincts) ayant même source et même but pouvaient coïncider en tant qu'opérateurs sur l'ensemble des listes de mots de type leur source sur un alphabet ordinaire quelconque. Nous allons voir maintenant que pareille coïncidence ne peut se présenter si l'on considère les mots sur un alphabet gradué contenant un nombre suffisant de signes distincts dans chacun des Λ_i (cf. 5.1).

6.1.1. *De façon précise*, on peut montrer que le schéma d'un morphisme $RO est déterminé par la donnée de deux listes, la première notée :

$$@\ X\ @$$

composée de mots-signes deux à deux différents (cf. 5.1.5) et telle que $RO puisse opérer sur elle; la seconde, transformée de @ X @ par $RO et notée :

$$@\ Y\ @ = \$RO(@\ X\ @).$$

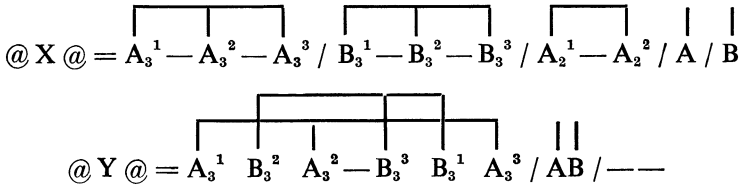
La source et le but du schéma du morphisme $RO ne sont autres que les schémas de @ X @ et @ Y @ respectivement. Quant au graphe, il est donné par les peignes-liens de @ Y @ dont chacun peut être identifié, dent pour dent, à un peigne de la source;

En effet, les peignes-schémas de @ X @ coïncident avec ses peignes-liens; par suite, si deux lettres de la matière (cf. 5.1.4), @ $\hat{Y}$ @, de @ Y @ ont même chemin (cf. 4.2), leurs origines respectives dans @ X @ sont reliées par un peigne-lien; et deux lettres de @ $\hat{Y}$ @ sont reliées par un peigne-lien (cf. 5.2) si et seulement si elles ont même chemin. Les peignes du graphe de $RO ne sont donc autres que les peignes-liens de @ Y @, en position. De plus,

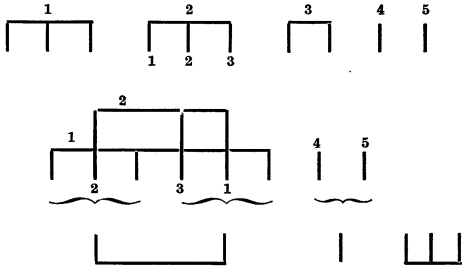
comme tous les mots-signes de @ X @ sont distincts entre eux, ils peuvent servir à distinguer les peignes les uns des autres : ainsi, on peut reconnaître quel peigne de la source correspond à un peigne du graphe, et, dans ce peigne de la source, quelle dent correspond à une dent du peigne du graphe considéré; de même, on peut reconnaître, grâce aux — et aux /, les groupes de dents du graphe à réunir d'une accolade au-dessus des dents du but.

On peut donc reconstituer le schéma complet du morphisme $RO.

Voici un exemple :



Le morphisme $RO qui transforme @ X @ en @ Y @ a pour schéma :

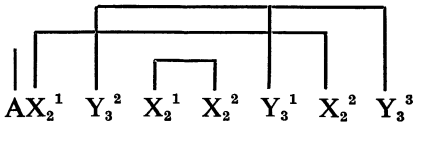


6.1.2. *Plus généralement,* étant donné deux listes @ X @ et @ Y @ ; si @ X @ est formée de mots-signes et comporte tous les signes de @ Y @, il existe un morphisme $RO tel que :

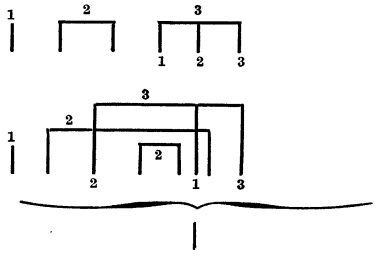
$$@\ Y\ @ = \$RO(@\ X\ @).$$

Si, de plus, les mots-signes de la liste @ X @ sont tous distincts, $RO est déterminé de façon unique.

Ainsi, tout mot sur un alphabet gradué Λ est le transformé par un morphisme d'une liste comprenant tous les mots-signes définis par les signes du mot considéré. Par exemple, le mot @ M @ :



est le transformé par le morphisme $RO de schéma :



de la liste :

$$A / X_2^{\,1} — X_2^{\,2} / Y_3^{\,1} — Y_3^{\,2} — Y_3^{\,3}$$

Posons :

$$@\,X\,@ = X_2^{\,1} — X_2^{\,2}$$

$$@\,Y\,@ = Y_3^{\,1} — Y_3^{\,2} — Y_3^{\,3}$$

le mot @ M @ n'est autre que :

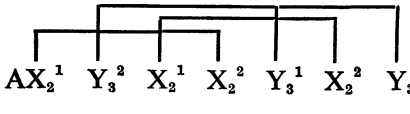
$$\$RO(A/@\ X\ @/@\ Y\ @)$$

où A/@ X @/@ Y @ désigne le produit de juxtaposition :

$$A / X_2^{\,1} — X_2^{\,2} / Y_3^{\,1} — Y_3^{\,2} — Y_3^{\,3}$$

des listes (chacune d'un seul mot-signe) A, @ X @ et @ Y @.

6.1.3. *Donc, ainsi qu'il a été dit au n° 5.1.7, un mot comportant* des signes à plusieurs insertions peut toujours être noté en langage de programmation comme résultat d'un morphisme agissant sur une liste d'éléments déjà reçus dans ce langage (cf. 5.1.7). Il y a une façon canonique de noter chaque mot comme transformée d'une liste-source formée des mots-signes définis par les signes du mot considéré (et d'eux seulement) pris une fois et une seule dans l'ordre où l'on rencontre pour la première fois leurs premières insertions quand on lit le mot de gauche à droite. e. g. ci-dessus, $RO(A/@ X @/@ Y @) est la façon canonique de noter le mot



6.2. Opérateurs définis à partir des morphismes.

Nous allons définir, à partir de morphismes, des opérateurs sur des ensembles de listes; nous verrons au n° 6.3 que les opérateurs définis ici sont des morphismes.

Dans ce numéro, Λ désigne un alphabet ordinaire ou un alphabet gradué.

Le / produit de deux morphismes (ou produit direct) : soient $RO et $MU deux morphismes de même source; S l'ensemble des listes de mot sur Λ de type la source de $RO et $MU. $RO et $MU sont deux opérateurs sur S. Le / produit de $RO et $MU, noté :

($RO/$MU)

est l'opérateur sur S ainsi défini, pour toute liste @ X @ de S :

($RO/$MU) (@ X @) = $RO(@ X @)/$MU(@ X @)

(i. e. la liste transformée par $RO/$MU de @ X @ est le produit de juxtaposition des listes transformées de @ X @ par $RO et $MU).

Le // produit de deux morphismes : soit $RO et $KHI deux morphismes; S l'ensemble des listes de mots sur Λ de type la source de $RO; T l'ensemble des listes de mots sur Λ de type la source de $KHI. $RO est un opérateur sur S, $KHI sur T. Notons S/T l'ensemble des listes de mots sur Λ produits de juxtaposition d'une liste de S par une liste de T. Le // produit de $RO et $KHI, noté :

($RO//$KHI)

est l'opérateur sur S/T ainsi défini :

pour toute liste @ X @/@ Z @ de S/T

($RO//$KHI) (@ X @/@ Z @) = $RO(@ X @)/$KHI(@ Z @).

Il est à noter que le // produit de deux morphismes est toujours défini, quels que soient ces deux morphismes, alors que le / produit de deux morphismes n'est défini que s'ils ont même source.

Le * produit de deux morphismes (ou composition) : soit $RO et $PI deux morphismes tels que la source de $RO soit identique au but de $PI; R l'ensemble des listes de mots sur Λ de type la source de $PI; S et U les ensembles des listes de mots sur Λ de type respectivement la source et le but de $RO. $PI est un opérateur sur R qui transforme toute liste de R en une liste de S ; $RO est un opérateur sur S qui transforme toute liste de S en une liste de U; en particulier, toute liste @ X @ de S transformée par $PI

d'une liste @ V @ de R est transformée par $RO en une liste @ W @ de U : nous dirons que @ W @ est la liste transformée de @ V @ par le * produit de $RO et $PI. De façon précise, le * produit de $RO et $PI noté :

($RO * $PI)

est l'opérateur sur R ainsi défini :

pour toute liste @ V @ de R,

($RO * $PI) (@ V @) = $RO($PI(@ V @)).

Notons que le * produit ($RO * $PI) des deux morphismes $RO et $PI n'est défini que si la source de $RO est identique au but de $PI; il transforme alors toute liste de type la source de $PI en une liste de type le but de $RO.

6.3. **Les opérateurs définis** en 6.2 comme / produit, // produit et * produit de morphismes sont des morphismes.

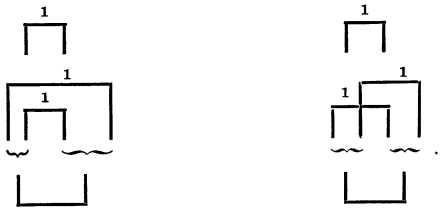
En effet, on peut les représenter par des schémas qui sont des schémas de morphismes.

Le schéma de ($RO/$MU) est ainsi construit :

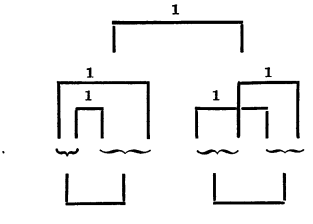
la ligne 1 est la source commune aux schémas de $RO et $MU;

les lignes 2 et 3 sont celles de $RO et $MU placées bout à bout, celles de $MU à la droite de celles de $RO. Par exemple, soit :

$RO : $MU :



Le schéma de ($RO/$MU) est le suivant :

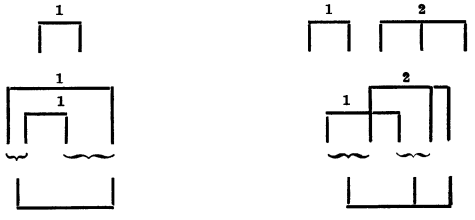


Le schéma de (\$RO//\$KHI) est ainsi construit :

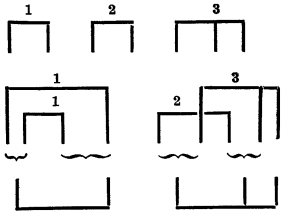
on modifie les numéros des peignes du schéma de \$KHI en ajoutant à chacun d'eux le nombre de peignes de la source de \$RO; puis on place les schémas de \$RO et de \$KHI bout à bout, celui de \$KHI à la droite de celui de \$RO.

Par exemple soit :

\$RO : \$KHI :



Le schéma de (\$RO//\$KHI) est le suivant :



Le schéma de (\$RO * \$PI) est ainsi construit :

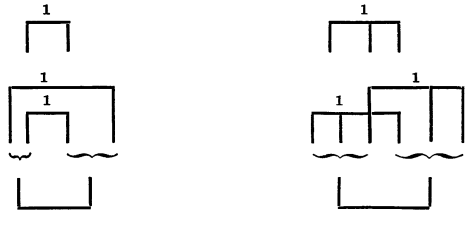
La source est celle de \$PI.

Le graphe est celui de \$RO où l'on remplace chaque peigne (qui est aussi un peigne du but de \$PI) par le système de peignes qui lui correspond dans le graphe de \$PI.

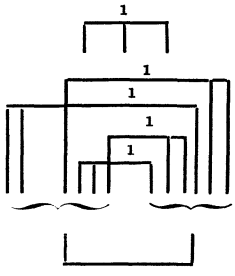
Le but est celui de \$RO.

Par exemple, soit :

\$RO : \$PI :

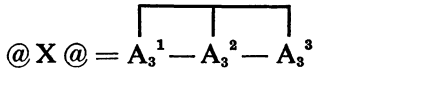


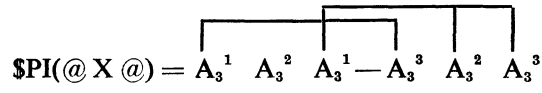
Le schéma de ($RO * $PI) est le suivant :



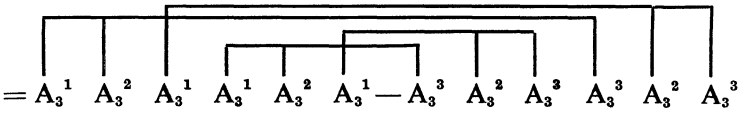
Pour construire le schéma du * produit de $RO et $PI, on peut aussi le caractériser par son action sur une liste @ X @, de type la source de $PI, composée uniquement de mots-signes deux à deux différents (cf. 6.1). Pour cela, nous ferons agir $PI sur @ X @, puis $RO sur $PI (@ X @).

Reprenons l'exemple précédent; soit :





$RO($PI(@ X @))



d'où l'on déduit le schéma de ($RO * $PI).

6.3.1. *Remarque.*

Les différents produits /, // et * définis entre les morphismes ne sont pas associatifs entre eux. En revanche, il y a associativité lorsque n'interviennent que des produits d'une seule sorte (/, // ou *), ce qui permet, dans ce cas, de supprimer les parenthèses. Ainsi :

$$(\$RO * \$PI) * \$XI) = (\$RO * (\$PI * \$XI))$$

pourra se noter :

$$\$RO * \$PI * \$XI$$

6.4. Échelles.

Correspondant aux notions formelles d'éléments non connexes (ou peignes-schémas), de types de listes (ou suites finies de peignes), de morphismes (définis par leurs schémas) nous avons défini sur tout alphabet Λ (alphabet ordinaire ou alphabet gradué) :

1° Les mots non connexes (de modèles les peignes) et les listes de mots non connexes (de modèles les types de listes) : à tout peigne correspond l'ensemble des mots non connexes de type ce peigne; à tout type de liste correspond l'ensemble des listes de ce type.

2° Les morphismes-opérateurs, définis par des morphismes schémas (pour les termes : morphismes-opérateurs, morphismes-schémas, cf. 4.6 *in fine*) : ce sont des applications ensemblistes; à tout morphisme-schéma correspond une application, bien définie, de l'ensemble de toutes les listes de type la source (du schéma donné) dans l'ensemble des listes de type le but.

Ce système d'ensembles (ensembles de listes) et d'applications entre ces ensembles constituent ce que nous appellerons l'échelle libre sur l'alphabet Λ.

Plus généralement, une échelle sera ainsi définie :

1° à tout peigne-schéma correspond un ensemble d'éléments abstraits, les pseudo-mots non connexes, de type ce peigne; de même, à tout type de liste correspond un ensemble d'éléments abstraits, les pseudo-listes, de ce type;

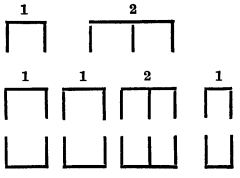
2° à tout morphisme-schéma correspond une application de l'ensemble des pseudo-listes de type la source (du morphisme) dans l'ensemble des pseudo-listes de type le but; de telle sorte que soient remplies les conditions suivantes :

a) Soit $RO et $PI deux morphismes-schémas tels que ($RO * $PI) soit défini (i. e. tels que la source de $RO soit identique au but de $PI).
Au morphisme-schéma ($RO * $PI) correspond (selon 2°) l'application composée des applications correspondant à $RO et $PI.

b) Tout ensemble de pseudo-listes de type donné est le produit des ensembles de pseudo-mots de types les peignes de la liste constituant le type des pseudo-listes considérées (autrement dit, toute pseudo-liste est une suite finie de pseudo-mots de type convenable).

Tout ensemble de pseudo-listes est donc muni naturellement de projections ensemblistes sur les ensembles de pseudo-mots dont il est le produit. Ces projections ensemblistes doivent être les applications correspondant

(selon 2°) aux morphismes projections élémentaires (cf. 4.3) que l'on peut définir sur le type de liste donné; et de même pour les projections généralisées e. g. à la projection généralisée de schéma :



correspond l'application qui à toute pseudo-liste de type la source associe la liste formée de deux fois le 1er pseudo-mot une fois le second, une fois le premier.

En bref, *a*) est une condition de compatibilité avec la composition des morphismes et *b*) avec les projections. De ces conditions résulte la compatibilité avec le / produit et le // produit des morphismes. Considérons, par exemple, le // produit de deux morphismes $RO et $KHI; soit x, y deux pseudo-listes de types, respectivement, les sources de $RO et $KHI, z la pseudo-liste obtenue en les mettant bout à bout. Il résulte de *a*) et *b*) que l'image de z par l'application associée (selon 2°) à ($RO//$KHI) est la pseudo-liste obtenue en mettant bout à bout les images de x et y par l'application associée à $RO et celle associée à $KHI respectivement.

7. LANGAGE DE PROGRAMMATION

Nous récapitulons ici les diverses notations signalées au cours de ce paragraphe en vue du traitement sur machine des problèmes de linguistiques. Dans nos descriptions des signes et assemblages du langage de programmation, nous utilisons, pour les désigner, les signes métalinguistiques σ, $\mathfrak{S}$, $\mathfrak{L}$, $\mathcal{M}$ (étrangers à ce langage).

7.1. **Les signes du langage de programmation** sont les suivants. D'une part,

— / @ $ ( ) * =

D'autre part,

les capitales latines
les chiffres
#
. ,

Nous notons ici σ l'un quelconque de ces derniers signes et $\mathfrak{E}$ une suite quelconque (éventuellement vide) de signe σ. C'est ce qu'indiquent les règles suivantes :

$$\sigma \rightarrow \{ A, B, \ldots, 0, 1, \ldots, 9, ., ., \neq \}$$
$$\mathfrak{E} \rightarrow \text{vide}$$
$$\mathfrak{E} \rightarrow \sigma\mathfrak{E}.$$

7.2. **Désignons par $\mathcal{M}$ l'expression d'un morphisme** (cf. n° 3 et 4) quelconque. $\mathcal{M}$ peut être :

un assemblage $\mathfrak{E}, symbole de morphisme (cf. n° 44),

un assemblage ($\mathcal{M}$ * $\mathcal{M}$) ou ($\mathcal{M}$/$\mathcal{M}$) ou ($\mathcal{M}$//$\mathcal{M}$) définissant le morphisme, par là exprimé, comme * produit, / produit ou // produit (cf. 6.2) de deux autres morphismes (ces derniers pouvant eux-mêmes être exprimés soit par $\mathfrak{E}, soit par ($\mathcal{M}$ * $\mathcal{M}$), etc.). D'où, en résumé, les règles suivantes opérant récursivement (un nombre quelconque de fois, l'une après l'autre) :

$$\mathcal{M} \rightarrow \$\,\mathfrak{E}$$
$$\mathcal{M} \rightarrow (\mathcal{M} * \mathcal{M})$$
$$\mathcal{M} \rightarrow (\mathcal{M}/\mathcal{M})$$
$$\mathcal{M} \rightarrow (\mathcal{M}//\mathcal{M}).$$

7.3. **Désignons par $\mathfrak{L}$ l'expression d'une liste** (cf. n° 2.2) quelconque. Rappelons qu'une liste peut être réduite à un seul mot (éventuellement vide ou réduit à un seul signe σ), que les symboles de listes sont de la forme @ $\mathfrak{E}$ @ (cf. 2.3), qu'un morphisme opérant sur une liste définit une nouvelle liste (cf. 4.2) et que l'on peut (cf. 2.4) désigner des listes en juxtaposant (avec interposition éventuelle d'un — ou d'un /) des expressions de types divers représentant des listes. $\mathfrak{L}$ sera donc formée en appliquant récursivement les règles suivantes :

$$\mathfrak{L} \rightarrow \text{vide}$$
$$\mathfrak{L} \rightarrow \sigma$$
$$\mathfrak{L} \rightarrow —$$
$$\mathfrak{L} \rightarrow /$$
$$\mathfrak{L} \rightarrow \mathfrak{L}\mathfrak{L}$$
$$\mathfrak{L} \rightarrow @\ \mathfrak{E}\ @$$
$$\mathfrak{L} \rightarrow \mathcal{M}(\mathfrak{L}).$$

REMARQUE. — En notant @ $\mathfrak{E}$ @ les signes à plusieurs insertions, le formalisme ci-dessus (cf. 6.1.3, 5.1.7) permet de noter les mots et listes de mots sur un alphabet gradué.

7.4. **A chaque fois que l'on introduira** dans le programme un symbole $\$\mathfrak{E}$, on indiquera quel morphisme il représente par une déclaration de morphisme ainsi écrite :

$$\$\mathfrak{E} = @ \ldots @ \$ \ldots \$$$

entre les deux signes @ se trouvera une liste dont toutes les composantes sont vides et entre les deux signes $ une suite de triples d'entiers séparés par des . ,—ou / (cf. n° 4.4).

7.5. **A chaque fois que l'on introduira** dans le programme un symbole $@\ \mathfrak{E}\ @$ on pourra soit indiquer quelle liste il représente par une déclaration de liste ainsi écrite :

$$@\ \mathfrak{E}\ @ = \mathfrak{L}$$

soit seulement indiquer de quel type est la liste $@\ \mathfrak{E}\ @$ par une déclaration de type de liste ainsi écrite :

$$@\ \mathfrak{E}\ @ = @ \ldots @$$

entre les deux signes *a* du membre de droite se trouvera une liste dont toutes les composantes sont vides (cf. 2.3) (i. e. une suite de—et de /).

7.6. **Les notations ci-dessus** peuvent être intégrées dans un système complet de programmation comprenant notamment :

une partie logique (les comparaisons, les branchements, les boucles...),
des ordres opératoires (par exemple : calculer la déclaration d'un morphisme exprimé en fonction d'autres morphismes, ou calculer une liste exprimée en fonction d'autres listes et de morphismes),
des instructions d'entrée-sortie.

Le système de macroinstructions, dû à H. Vandenburgh, ne comporte que des ordres opératoires et des instructions d'entrée-sortie.

(Manuscrit reçu le 8 juillet 1966)