MÉMOIRES DE LA S. M. F.

JEAN-FRANÇOIS PERROT

Utilisation d'APL pour calculer des monoïdes finis

Mémoires de la S. M. F., tome 49-50 (1977), p. 159-176

<http://www.numdam.org/item?id=MSMF_1977__49-50__159_0>

Calculateurs en Math. (1975 - LIMOGES)
Bull. Soc. math. France
Mémoire 49-50, 1977, p. 159-176

UTILISATION D'APL POUR CALCULER DES MONOÏDES FINIS

par Jean-François PERROT

Cet exposé a pour but de montrer, sur un exemple simple, comment un outil informatique adéquat peut rendre au mathématicien des services quotidiens : il ne s'agit pas ici de programmes exceptionnels utilisant l'ordinateur jusqu'aux limites de sa capacité de mémoire, ou destinés à ne servir que rarement (pour établir ou réfuter une conjecture, par exemple), mais d'instruments permettant d'effectuer commodément des calculs fastidieux.

En l'occurrence, ces calculs décrivent la structure d'un monoïde fini M, représenté par transformations d'un ensemble, à partir d'un système générateur de M. Ce système générateur peut être interprété comme la fonction de transitions d'un automate fini dont M est alors le monoïde de transitions. Lorsque cet automate est l'automate minimal (ou réduit) reconnaissant un langage L (cf. [8], [10] [13]) M est isomorphe au monoïde syntactique de L : c'est afin de calculer les monoïdes syntactiques de certains langages particuliers que nous avons commencé à écrire des programmes APL [15].

Ce calcul le plus simple et le plus naturel est celui de l'ensemble des éléments de M, que l'on obtient par énumération à partir du système générateur. On en tire aisément la table de multiplication et le graphe de Cayley. Nous donnons ci-dessous, à titre d'illustration, une manière d'effectuer ce travail en APL.

Le calcul exhaustif devient impraticable pour les monoïdes à plusieurs milliers d'éléments que nous avons couramment rencontrés dans nos recherches, et avec lui les méthodes proposées par différents auteurs pour obtenir les renseignements désirés sur la structure de M ([2] [4] [11]). Ces renseignements touchent en général la structure des idéaux de M et se trouvent commodément résumés dans les équivalences de Green. Il est donc souhaitable d'avoir accès à la structure des classes de Green sans énumérer tous les éléments qui les composent : c'est essentiellement ce que permet, sous certaines hypothèses restrictives, la méthode que nous avons présentée en [15], développement d'une idée due à M.P. Schützenberger.

Cette méthode a en outre l'avantage d'être assez souple pour s'appliquer à différents problèmes particuliers où les renseignements cherchés ne touchent qu'une partie du monoïde, ce qui autorise des économies de calcul substantielles : nous donnons ici, comme exemple d'application de cette méthode, le calcul de l'idéal minimal du monoïde M.

On trouvera en [6] une autre application, le calcul complet d'un monoïde régulier.

Quant à l'outil informatique dont les qualités sont si bien adaptées à notre problème, c'est le système conversationnel APL disponible sur IBM 360. On sait qu'APL a d'abord été, plutôt que "A Programming Language" [12], un ensemble de notations permettant de traduire directement des organigrammes sous forme séquentielle. Délaissant les structures de contrôle - boucles FØR, WHILE, REPEAT, etc.- et les règles de transmission de paramètres dans les appels de procédures mises au point pour les langages de la famille Algol et qui intéressent si fort les théoriciens de l'informatique, Iverson a choisi de construire, à grand renfort de lettres grecques, de caractères gras et de signes diacritiques, un jeu d'opérateurs de base très élaborés traitant directement les données structurées, tableaux à une ou plusieurs dimensions (vecteurs, matrices, etc.) et arborescences. Malgré les exemples d'applications de sa notation à des problèmes variés que propose l'auteur, cette tentative de formalisation des algorithmes serait probablement restée sans suite si elle n'avait été mise en oeuvre avec une remarquable efficacité sur les ordinateurs IBM de la série 360. Les problèmes typographiques ont été résolus avec une élégance peu commune grâce à la technologie des machines à écrire à boules interchangeables : la "boule APL" porte un jeu de 88 caractères soigneusement dessinés, très agréables à l'oeil, permettant de réaliser sur le terminal IBM 2741 une version simplifiée de la notation d'origine (cf. [9]). Par l'intermédiaire du terminal, l'ordinateur se comporte comme une machine de bureau taillable et corvéable à merci : chaque utilisateur règne sans partage sur son "espace de travail" (cf. [13]), la machine répond immédiatement à ses requêtes (exemples ci-dessous). Et l'hermétisme redoutable de la notation devient, pour qui doit taper ses programmes lui-même au clavier, une concision pleine de charme (nombreux exemples dans [17]).

Le succès remporté par APL/360 auprès d'utilisateurs variés explique les développements ultérieurs du produit : en dehors d'IBM, il existe maintenant des systèmes APL, certains excellents, sur de nombreuses machines, notamment sur de petits ordinateurs comme le T 1600 de la Télémécanique (cf. [7] pour une description générale avec bibliographie, et [1] pour une tentative orientée vers les mathématiques). On trouvera dans les exposés de J. Michel et de G. Viennot à ces mêmes Journées d'autres exemples d'utilisation de ce remarquable outil.

Pour toutes les notions classiques relatives à la théorie des monoïdes, (qui ne diffèrent des demi-groupes - en anglais, semi groups - que par la présence obligatoire d'un élément neutre) le lecteur est renvoyé à l'ouvrage de Clifford et Preston [5].

I - NOTATIONS FONDAMENTALES

L'ensemble fini S à N éléments sur lequel opère le monoïde objet du calcul sera toujours identifié avec l'ensemble des entiers de 1 à N et représenté en machine par le vecteur ιN = 1, 2, ..., N -1 (l'origine des indices étant 1). Les transformations de S sont notées à droite, de façon à ce que, pour $s \in S$ et $p, q \in S^S$, on ait spq = (sp)q ; une transformation p de S est alors représentée en machine par un vecteur P à N composantes, la $i^{ème}$ composante P [I] valant ip ; dans ces conditions, si les vecteurs P1 et P2 représentent les transformations p_1 et p_2, le produit p_1p_2 est représenté par P2 [P1]. Ainsi, par exemple :

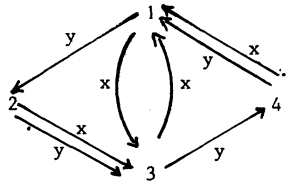
```
      N ←5
      P1←2 5 4 4 1
      P2←2 3 1 5 4
      P1[P2]
5  4  2  1  4
      P2[P1]
3  4  5  5  2
```

Un ensemble de k transformations de S est donc représenté par un tableau à k lignes et N colonnes, la $j^{ème}$ ligne du tableau représentant la $j^{ème}$ transformation de l'ensemble. En particulier, le système générateur du monoïde considéré est logé dans un tableau GENE à Q lignes.

Par exemple, avec Q = 2, le système GENE1 suivant engendre le groupe symétrique de degré 3 :

```
   GENE1
2  3  1
1  3  2
```

La fonction de transitions du semi-automate à quatre états, ayant pour alphabet d'entrée X = {x, y}, que voici, est représentée par GENE2 :




```
   GENE2
3  3  1  1
2  3  4  1
```

II - ENUMERATION

a) Le processus énumératif

On construit progressivement un tableau LISTE qui au début se réduit à une seule ligne, égale à ιN -i. e. la transformation identique, élément du monoïde - et à la fin contient l'ensemble des éléments du monoïde engendré par GENE.

Différentes variantes (cf. [4], [13]) quant à l'ordre des opérations se ramènent toutes à opérer successivement à droite toutes les lignes de LISTE par toutes les lignes de GENE : lorsqu'une de ces multiplications donne naissance à un élément qui n'est pas encore répertorié dans LISTE il est adjoint à ce dernier tableau; le processus s'arrête quand toutes les lignes de LISTE ont été opérées de la sorte sans plus donner de nouveaux éléments.

Ne pas devoir déclarer a priori le nombre de lignes du tableau LISTE est clairement un avantage. De plus, les opérations décrites ci-dessus s'énoncent très simplement en APL :

- multiplication de la $L^{ième}$ ligne de LISTE par la $X^{ième}$ ligne de GENE :

```
P ← LISTE [L ;]
PX ← GENE [X ; P]
```

- PX ne figure pas dans LISTE ssi l'expression V/LISTE ∧. = PX prend la valeur 0;
- ajouter PX à LISTE : LISTE ← LISTE, [1] PX.

Ainsi par exemple :

```
      GENE
3 3 1 1
2 3 4 1
      LISTE
1 2 3 4
3 3 1 1
2 3 4 1
1 1 3 3
4 4 2 2
3 1 1 3
3 4 1 2
2 2 4 4
1 3 3 1
4 2 2 4
      L
7
      X
1
      P←LISTE[L;]
      P
3 4 1 2
      PX←GENE[X;P]
      PX
1 1 3 3
      ∨/LISTE∧.=PX
1
      X←2
      PX←GENE[X;P]
      PX
4 1 2 3
      ∨/LISTE∧.=PX

0
```

```
      LISTE,[1]PX
1  2  3  4
3  3  1  1
2  3  4  1
1  1  3  3
4  4  2  2
3  1  1  3
3  4  1  2
2  2  4  4
1  3  3  1
4  2  2  4
4  1  2  3
```

b) Programme

La fonction APL suivante prend pour valeur le nombre d'éléments du monoïde engendré par GENE; son exécution provoque le rangement dans le tableau LISTE de l'ensemble des éléments du monoïde.

```
      ∇ENUMERER[□]∇
    ∇ NB←ENUMERER GENE
[1]   Q←(ρGENE)[1]
[2]   N←(ρGENE)[2]
[3]   LISTE←(1,N)ρP←ιN
[4]   L←LL←X←1
[5]  ITER1:→ITER2×ι∨/LISTE∧.=PX←GENE[X;P]
[6]   LL←LL+1
[7]   LISTE←LISTE,[1] PX
[8]  ITER2:→ITER1×ιQ≥X←X+1
[9]   →FIN×ιL≥LL
[10]  →ITER1,P←LISTE[L←L+X←1;]
[11] FIN:NB←L
    ∇

      ENUMERER GENE1
6
      LISTE1←LISTE
      LISTE1
1  2  3
2  3  1
1  3  2
3  1  2
3  2  1
2  1  3
```

```
      ENUMERER GENE2
12
      LISTE2←LISTE
      LISTE2
 1  2  3  4
 3  3  1  1
 2  3  4  1
 1  1  3  3
 4  4  2  2
 3  1  1  3
 3  4  1  2
 2  2  4  4
 1  3  3  1
 4  2  2  4
 4  1  2  3
 2  4  4  2
```

c) Table de multiplication

Si on numérote de 1 à M = ENUMERER GENE les éléments du monoïde, la loi de composition est donnée par la fonction :

```
      ∇MULT[□]∇
    ∇ PROD←I MULT J
[1]   PROD←(LISTE∧.=LISTE[J;LISTE[I;]])⍳1
    ∇

      ENUMERER GENE1
6
      1 MULT 5
5
      3 MULT 3
1

      ENUMERER GENE2
12
      2 MULT 10
8
      3 MULT 11
1
      10 MULT 2
9
```

A défaut d'un produit extérieur sur MULT, une double boucle permet de calculer la table de multiplication du monoïde, dont l'impression ne sera demandée que si M n'est pas trop grand ! On observera que 1 est toujours l'élément neutre du monoïde, de sorte que la première ligne et la première colonne de la table répètent l'énumération des éléments.

```
      ∇TABLE[□]∇
    ∇ TABLE GENE
[1]   M←ENUMERER GENE
[2]   OP←⍳I←J←0
[3]   I←I+J←1
[4]   OP←OP,I MULT J
[5]   →4×⍳M≥J←J+1
[6]   →3×⍳M>I
[7]   OP←(M,M)⍴OP
    ∇
```

```
    TABLE GENE1
    OP
1  2  3  4  5  6
2  4  5  1  6  3
3  6  1  5  4  2
4  1  6  2  3  5
5  3  2  6  1  4
6  5  4  3  2  1
```

```
    TABLE GENE2
    OP
 1   2   3   4   5   6   7   8   9  10  11  12
 2   4   5   2   8   4   4   5   2   8   8   5
 3   6   7   9  10   4  11  12   2   8   1   5
 4   2   8   4   5   2   2   8   4   5   5   8
 5   4   4   2   8   2   8   5   4   5   2   8
 6   9  10   6  12   9   9  10   6  12  12  10
 7   4  11   2   8   9   1   5   6  12   3  10
 8   2   2   4   5   4   5   8   2   8   4   5
 9   6  12   9  10   6   6  12   9  10  10  12
10   9   9   6  12   6  12  10   9  10   6  12
11   9   1   6  12   2   3  10   4   5   7   8
12   6   6   9  10   9  10  12   6  12   9  10
```

d) Graphe de Cayley

La table de multiplication pouvant être fort encombrante, certains [4] [13] préfèrent représenter le monoïde par son graphe de Cayley, qui a pour sommets les éléments du monoïde et, pour tout élément x du système générateur, un arc étiqueté x allant de m_1 à m_2 (m_1 , m_2 éléments du monoïde) si $m_1 x = m_2$.

Le système générateur étant repéré dans LISTE par le vecteur d'indices IND, la fonction suivante calcule le graphe de Cayley sous forme d'un tableau à M lignes et Q + 1 colonnes, chaque ligne contenant le numéro d'un sommet (= élément du monoïde) et ceux de ses successeurs correspondant aux Q arcs issus de lui.

```
      ∇CAYLEY[□]∇
    ∇ GPH←IND CAYLEY LISTE
[1]   QI←ρIND
[2]   M←(ρLISTE)[1]
[3]   GPH←ιL←0
[4]   GPH←GPH,L←L+X←1
[5]   GPH←GPH,L MULT IND[X]
[6]   →5×ιQI≥X←X+1
[7]   →4×ιM>L
[8]   GPH←(M,QI+1)ρGPH
    ∇
```

```
      (2 3) CAYLEY LISTE1
1  2  3
2  4  5
3  6  1
4  1  6
5  3  2
6  5  4
```

```
      (2 3) CAYLEY LISTE2
 1   2   3
 2   4   5
 3   6   7
 4   2   8
 5   4   4
 6   9  10
 7   4  11
 8   2   2
 9   6  12
10   9   9
11   9   1
12   6   6
```

III. CALCUL DE L'IDEAL MINIMAL

a) Définition

Une partie R (resp. L, resp. D) d'un monoïde M vérifiant R M = R (resp. M L = L, resp. M D M = D) est un idéal à droite (resp. à gauche, resp. bilatère) de M. Un idéal non vide qui ne contient strictement aucun autre idéal non vide de la même espèce est dit minimal. Il est clair que tout monoïde fini possède des idéaux minimaux à droite et à gauche et un seul idéal bilatère minimal, qui est l'intersection de tous ses idéaux bilatères non vides : ce dernier est parfois appelé noyau du monoïde fini ("kernel" ou "Kerngruppe"). On démontre à ce sujet le résultat suivant (cf. [5]).

Soient $\{R_i\}_{i=1}^{k}$ et $\{L_j\}_{j=1}^{h}$ les familles des idéaux minimaux, à droite et à gauche respectivement, du monoïde fini M, et J son idéal bilatère minimal. On a $J = \bigcup_{i=1}^{k} R_i = \bigcup_{j=1}^{h} L_j$.

Nous appelerons désormais J l'idéal minimal de M. C'est à la détermination de sa structure qu'est consacrée la fin de cet exposé.

Signalons que l'idéal minimal joue un rôle prépondérant en théorie des automates fortement connexes (cf. [14], [16]) et par suite dans la théorie des codes préfixes complets. La technique de calcul que nous allons donner est donc un outil essentiel pour l'étude approfondie de ces théories.

Posons $H_{ij} = R_i \cap L_j$. On vérifie que H_{ij} est fermé vis à vis de la multiplication du monoïde M, et que cette opération lui donne une structure de groupe : H_{ij} contient donc un élément idempotent de M et un seul, l'élément neutre du groupe. De plus, tous les groupes H_{ij} ($i = 1, \ldots, k$, $j = 1, \ldots, h$) sont isomorphes deux à deux. Cette observation conduit à représenter l'idéal minimal J par un diagramme rectangulaire en "boîte à oeufs", dont les lignes sont les k idéaux à droites minimaux R_i, les colonnes les h idéaux à gauche minimaux L_j et les cases individuelles les hk groupes isomorphes H_{ij} (cf. infra).

Il est clair qu'avec $p \in J$ et $q \in J$ on a encore $pq \in J$, de sorte que J forme naturellement un demi-groupe (sans élément unité, autrement il se réduit à un groupe) dont la structure est donnée par la variante suivante du théorème de Rees-Suschkewitsch [5] :

Soit $G = H_{11}$ le groupe en première ligne, première colonne du diagramme en boîte à oeufs précédent; considérons la matrice Γ à éléments dans G, de dimensions $h \times k$, d'éléments courant γ_{ji}, avec $\gamma_{ji} = e_{1j}\, e_{i1}$, où e_{1j} désigne l'unique idempotent contenu dans $R_1 \cap L_j$ et e_{i1} son homologue de $R_i \cap L_1$; alors J est isomorphe au demi-groupe obtenu en munissant l'ensemble $\{1, 2, 3, \ldots, k\} \times G \times \{1, 2, 3, \ldots, h\}$ de la multiplication $(i', g', j') \cdot (i'', g'', j'') = (i', g'\, \gamma_{j'i''}\, g'', j'')$.

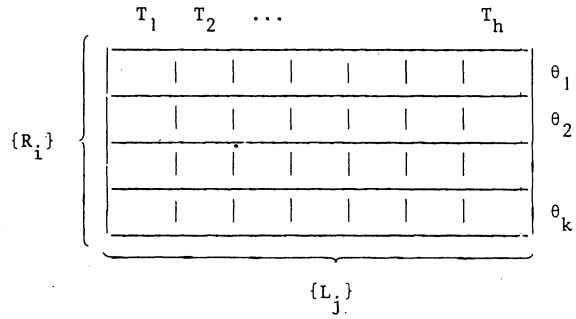
En somme, la détermination de la structure de J se ramène au calcul de la matrice Γ, dite matrice-sandwich, et à l'identification du groupe G. C'est à cette tâche que nous nous employons maintenant.

b) Images, équivalences, rang

Pour toute application $p \in S^S$, désignons par Im(p) le sous-ensemble image $Sp \subset S$, par Ker(p) l'équivalence d'application de p et par rg(p) le rang de p, qui est égal à l'index de Ker(p) et au cardinal de Im(f). On vérifie facilement que l'idéal minimal J est exactement l'ensemble des éléments de M de rang minimum.

Dans l'exemple engendré par GENE2, ce rang minimum est 2, et J est formé des éléments n° 2, 4, 5, 6, 8, 9, 10 et 12.

De plus, les idéaux à gauche (resp. à droite) minimaux sont formés de tous les éléments de J ayant la même image (resp. la même équivalence) de sorte que les colonnes du diagramme en boîte à oeufs sont indexées par les différents ensembles-images minimaux T_1 , ..., T_h et les lignes par les équivalences θ_1 , ..., θ_k associées.



Enfin, chaque groupe H_{ij} apparaît comme groupe de permutations sur l'ensemble image T_j . Tous ces groupes de permutations sont équivalents au groupe de structure de l'idéal minimal J.

Ainsi, dans notre exemple, on obtient :

	$T_1 = \{1, 3\}$	$T_2 = \{2, 4\}$
$\theta_1 = 12/34$	2, 4	5, 8
$\theta_2 = 14/23$	6, 9	10, 12

(les numéros soulignés sont ceux des idempotents, éléments neutres des sous-groupes).

Le groupe de structure est en ce cas le groupe symétrique sur deux éléments.

c) Calcul de la matrice-sandwich

Le calcul de Γ requiert seulement la connaissance des idempotents de la première ligne et de ceux de la première colonne : or, un idempotent e est entièrement déterminé si on connaît son ensemble-image $T = Im(e)$ et son équivalence d'application θ, car les éléments de T doivent former un système complet de représentants des classes mod θ, et e envoie chaque classe mod θ sur son unique représentant dans T.

Dans l'exemple engendré par GENE2, on obtient ainsi $e_{11} = 1133$, $e_{12} = 2244$, $e_{21} = 1331$, et $\Gamma = \begin{pmatrix} \varepsilon & \varepsilon \\ \varepsilon & \alpha \end{pmatrix}$, où ε est la permutation identique et α la transposition (1, 3).

On peut remarquer à ce sujet que les éléments de la première ligne et de la première colonne de Γ sont tous égaux à la permutation identique.

Le calcul de Im(p) est effectué par la fonction APL suivante :

```
      ∇IM[□]∇
    ∇ T←IM P
[1]   T←(IN∈P)/IN←⍳N
    ∇

      P
2  4  4  2
      IM P
2  4
      PX
3  1  1  3
      IM PX
1  3
```

Le rang de l'application p, à savoir |Im(p)|, est alors égal à ρ IM P.

Etant donné une transformation p et un ensemble image T tels que l'équivalence d'application de p et l'ensemble T déterminent un idempotent, ce dernier est calculé par la fonction :

```
      ∇IDP[□]∇
    ∇ PI←P IDP T
[1]   PI←(P∘.=P[T])+.×T
    ∇

      P
2  4  4  2
      P IDP 1 3
1  3  3  1
      PX
3  1  1  3
      PX IDP 1 2
1  2  2  1
      PX IDP 1 3
1  3  3  1
```

A partir d'un tableau LIDP à k = LK lignes et N colonnes, contenant les idempotents de la $1^{ère}$ colonne du diagramme en boîte à oeufs, et d'un tableau LIM à h = LI lignes et R colonnes, où R est le rang des éléments de J, contenant les images minimales $\{T_j ; j = 1, \ldots, h\}$, avec TH = T_1 , la fonction suivante, qui fait appel à la fonction IDP, calcule la matrice-sandwich Γ (le nom de cette fonction provient de ce qu'un idéal minimal est parfois appelé un "paragroupe").

```
      ∇PARAGROUPE[□]∇
    ∇ MAT←LIDP PARAGROUPE LIM
[1]   LIGNE←⍳0
[2]   LP←1
[3]   RTH←LIDP[1;]
[4]  BOUCLEL:LL←0
[5]  CONSTRUIRE:LIGNE←LIGNE,LIDP[LL←LL+1;(RTH IDP LIM[LP;])[TH]]
[6]   →CONSTRUIRE×⍳LL<LK
[7]   →BOUCLEL×⍳LI≥LP←LP+1
[8]   MAT←(LI,LK,R)⍴LIGNE
    ∇
```

La matrice-sandwich est calculée sous la forme d'un tableau à trois dimensions (cf. instruction 8) est imprimée comme telle, i. e. d'abord les permutations de la première ligne de la matrice, puis celles de la $2^{ème}$ ligne, etc... Avec l'exemple engendré par GENE2, $T_1 = \{1, 3\}$, on obtient comme résultat

```
1 3
1 3

1 3
3 1
```

d) La représentation ergodique

Quant au groupe de structure G, on peut en obtenir un système générateur en calculant ce que M.P. Schützenberger a appelé la représentation ergodique droite (appelée en [5] représentation de Schützenberger) du monoïde M, qui à chaque élément $m \in M$ associe une matrice $\mu(m)$, monomiale en ligne, à éléments dans $G \cup \{0\}$, de dimension $h \times h$, ainsi définie : $\mu(m)_{j,j'} = 0$ lorsque $L_j.m \neq L_{j'}$, et lorsque $L_j.m = L_{j'}$, $(\mu(m))_{j,j'}$ est la permutation de T_1, restriction à T_1 de $e_{1j}\, m_{11}$. L'ensemble des éléments non nuls de la famille $\{\mu(x)$; x dans le système générateur de M$\}$ forme un système générateur de G.

Dans l'exemple engendré par GENE2, le calcul donne, avec x et y pour générateurs :

$$\mu(x) = \begin{pmatrix} \alpha & o \\ \alpha & o \end{pmatrix} \qquad \mu(y) = \begin{pmatrix} o & \varepsilon \\ \alpha & o \end{pmatrix}$$

Il est clair que l'ensemble $\{\mu(x)$; x dans le système générateur de M$\}$ se calcule directement à partir de GENE et de l'ensemble $\{T_j$; j = 1, ..., h$\}$ des images minimales. Avec les mêmes données que la fonction PARAGROUPE du paragraphe précédent, i. e. les tableaux LIM, à LI lignes, et LIDP, et en utilisant le tableau GENE et la fonction IDP, la fonction suivante calcule l'ensemble $\{\mu(x)$; x dans le système générateur$\}$; chaque matrice $\mu(x)$ est imprimée en donnant successivement sur une même ligne, pour les différents idéaux à gauche minimaux

L_j , l'indice j, l'indice j' de L_j . x, puis 0, suivi de la permutation $(\mu(x))_{j,j'}$ correspondante :

```
      ∇ERGO[□]∇
    ∇ RDTE←LI ERGO LIM
[1]   DTE←⍳0
[2]   X←1
[3]   RTH←LIDP[1;]
[4]  BOUCLEX:L←0
[5]  OPERER:TX←IM GENE[X;T←LIM[L←L+1;]]
[6]   DTE←DTE,L,((LIM∧.=TX)⍳1),0,(RTH[GENE[X;RTH IDP T]])[TH]
[7]   →OPERER×⍳LI>L
[8]   →BOUCLEX×⍳Q≥X←X+1
[9]   RDTE←(Q,LI,R+3)⍴DTE
    ∇
```

Le résultat obtenu ci-dessus est donc imprimé ainsi :

```
1 1 0 3 1
2 1 0 3 1

1 2 0 1 3
2 1 0 3 1
```

e) Le calcul proprement dit -

Il résulte de ce qui précède que le calcul complet de la structure de l'idéal minimal J se ramène à la construction des deux tableaux LIM contenant les images minimales, qui indexent les idéaux à gauche minimaux, et LIDP contenant les idempotents de la première colonne du diagramme en boîte à oeufs, en bijection avec les idéaux à droite minimaux.

Le calcul de LIM se fait dans les 14 premières instructions de la fonction IDMIN ci-dessous par un processus exhaustif assez semblable à celui qui permet d'énumérer tous les éléments du monoïde (cf. II a) et b)), compliqué par un test sur le rang des éléments (pour arriver au rang minimum) utilisant le fait que tout élément de l'idéal minimal doit avoir même rang que son carré (instruction 6). Celui de LIDP est effectué par les instructions 15 à 23; il est plus simple et utilise essentiellement la fonction IDP.

Enfin, l'appel des fonctions ERGO et PARAGROUPE, en fournissant un système générateur du groupe de structure et la matrice-sandwich, achève la détermination de la structure de J.

```
      ∇IDMIN[□]∇
    ∇ IDMIN GENE
[1]   Q←(ρGENE)[1]
[2]   N←(ρGENE)[2]
[3]   PP←T←ιR←N
[4]  INITIO:LIM←(1,R)ρT
[5]   L←LL←X←1
[6]  ITER1:→RANGER×ιR=RR←ρT←IM PX←PX[PX←GENE[X;PP]]
[7]   R←RR
[8]   →INITIO,PP←PX
[9]  RANGER:→ITER2×ι∨/LIM∧.=T
[10]  LL←LL+1
[11]  LIM←LIM,[1] T
[12] ITER2:→ITER1×ιQ≥X←X+1
[13]  →SUITE×ιL≥LL
[14]  →ITER1,PP←PP IDP LIM[L←L+X←1;]
[15] SUITE:LI←L
[16]  LIDP←(1,N)ρPP←PP IDP TH←LIM[1;]
[17]  L←LL←X←1
[18] ITER3:→ITER4×ι∨/LIDP∧.=XP←PP[GENE[X;]] IDP TH
[19]  LL←LL+1
[20]  LIDP←LIDP,[1] XP
[21] ITER4:→ITER3×ιQ≥X←X+1
[22]  →FIN×ιL≥LL
[23]  →ITER3,PP←LIDP[L←L+X←1;]
[24] FIN:LK←L
[25]  'RANG'
[26]  R
[27]  'FORMAT'
[28]  □←FORMAT←LK,LI
[29]  'LE GROUPE DE STRUCTURE EST REPRESENTE SUR'
[30]  TH
[31]  'REPRESENTATION ERGODIQUE DROITE'
[32]  LI ERGO LIM
[33]  'MATRICE-SAUVWICH'
[34]  LIDP PARAGROUPE LIM
    ∇
```

Dans les exemples qui suivent, nous donnerons, après les résultats imprimés par IDMIN, les contenus des tableaux LIM et LIDP.

```
        IDMIN GENE2
RANG
2
FORMAT
2  2
LE GROUPE DE STRUCTURE EST REPRESENTE SUR
1  3
REPRESENTATION ERGODIQUE DROITE
  1  1  0  3  1
  2  1  0  3  1

  1  2  0  1  3
  2  1  0  3  1
MATRICE-SANDWICH
  1  3
  1  3

  1  3
  3  1


     LIM
  1  3
  2  4
     LIDP
  1  1  3  3
  1  3  3  1
```

Autre exemple, avec N = 16 :

```
     GENE
  2   3   9   9   3   9   2   3  10  11   1   1  11   1  10  11
  5   9   4   9   6   7   8   4  13   1  12   1  14  15  16  12

     IDMIN GENE
RANG
6
FORMAT
7  7
LE GROUPE DE STRUCTURE EST REPRESENTE SUR
1  2  3  9  10  11
REPRESENTATION ERGODIQUE DROITE
   1   1   0   2   3   9  10  11   1
   2   1   0   2   3   9  10  11   1
   3   1   0   2   3   9  10  11   1
   4   1   0   2   3   9  10  11   1
   5   1   0   2   3   9  10  11   1
   6   1   0   2   3   9  10  11   1
   7   1   0   2   3   9  10  11   1

   1   2   0   2   9   3  10   1  11
   2   3   0   2   3   9  10  11   1
   3   4   0   2   3   1  10  11   9
   4   5   0   2   3   1  10  11   9
   5   6   0   2   3   1  10  11   9
   6   7   0   2   3   9  10  11   1
   7   2   0   2   3   9  10  11   1
```

MATRICE-SANDWICH

1	2	3	9	10	11
1	2	3	9	10	11
1	2	3	9	10	11
1	2	3	9	10	11
1	2	3	9	10	11
1	2	3	9	10	11
1	2	3	9	10	11
1	2	3	9	10	11
1	3	2	9	11	10
1	11	2	9	3	10
1	11	2	9	3	10
1	11	2	9	3	10
1	3	2	9	11	10
1	3	2	9	11	10
1	2	3	9	10	11
1	3	10	9	11	2
1	11	10	9	3	2
1	11	10	9	3	2
1	11	2	9	3	10
1	3	2	9	11	10
1	3	2	9	11	10
1	2	3	9	10	11
1	3	10	9	11	2
1	11	10	9	3	2
9	11	10	1	3	2
9	11	2	1	3	10
9	3	2	1	11	10
1	3	2	9	11	10
1	2	3	9	10	11
1	3	10	9	11	2
1	3	10	9	11	2
9	3	10	1	11	2
9	3	2	1	11	10
9	3	2	1	11	10
1	3	2	9	11	10
1	2	3	9	10	11
1	3	2	9	11	10
1	3	2	9	11	10
9	3	2	1	11	10
9	3	2	1	11	10
9	3	2	1	11	10
1	3	2	9	11	10
1	2	3	9	10	11
1	3	2	9	11	10
1	3	2	9	11	10
1	3	2	9	11	10
1	3	2	9	11	10
1	3	2	9	11	10
1	3	2	9	11	10

```
      LIM
1     2    3    9   10   11
1     4    5    9   12   13
1     5    6    9   13   14
5     6    7   13   14   15
6     7    8   14   15   16
4     7    8   12   15   16
1     4    8    9   12   16

      LIDP
1     2    3    3    2    3    1    2    9   10   11   11   10   11    9   10
1     2    3    2    3   10    1    3    9   10   11   10   11    2    9   11
1     2    3    2   11   10    1    3    9   10   11   10    3    2    9   11
1     2    3    2   11   10    9    3    9   10   11   10    3    2    1   11
1     2    3    2   11    2    9    3    9   10   11   10    3   10    1   11
1     2    3    2    3    2    9    3    9   10   11   10   11   10    1   11
1     2    3    2    3    2    1    3    9   10   11   10   11   10    9   11
```

Pour ce dernier exemple, l'examen de la représentation ergodique montre que le groupe de structure est engendré par :

$$\beta_1 = (1, 2, 3, 4, 10, 11)$$
$$\beta_2 = (1, 2, 9, 10)$$
$$\beta_3 = (1, 2, 3)\ (9, 10, 11) = \beta_2^2\ \beta_1^4$$

Ces trois permutations laissent invariante la partition d'imprimitivité 1, 9/2, 10/3, 11 ; G a donc un quotient de degré 3 qui contient un cycle de longueur 3, image de β_1, et une transposition, image de β_2 : ce quotient est donc le groupe symétrique $\underline{S}_3$ entier. Le noyau de l'homomorphisme contient $\beta_2^2 = (1, 9)\ (2, 10)$ et $\beta_1^3 = (1, 9)\ (2, 10)\ (3, 11)$ qui donnent toutes les permutations de type (i, j), $(i, j)\ (k, l)$ et $(i, j)\ (k, l)\ (m, n)$, donc un noyau d'ordre 8. G est donc d'ordre 48, isomorphe au produit en couronne $\underline{S}_2 \wr \underline{S}_3$. Le nombre total d'éléments de l'idéal minimal J est exactement de $7 \times 7 \times 48 = 2352$

Conclusion

Les calculs précédents peuvent bien entendu être programmés en n'importe quel langage, et seules des raisons de commodité peuvent conduire à utiliser APL. On notera à cet égard que le texte même des programmes traduit directement la démarche algorithmique (que l'on songe à ce que donneraient des programmes Algol ou Fortran), ce qui permet de les modifier au gré des besoins (demande d'informations supplémentaires, ou suppression de calculs inutiles dans un cas particulier donné). L'exécution en mode conversationnel donne tout son intérêt à cette souplesse d'utilisation, on a ainsi affaire à un instrument de calcul d'usage quotidien.

BIBLIOGRAPHIE

(1) BRAFFORT P. et MERISSET-COFFINIERES P. Eds.- Le projet LIMA, in APLASM 73.- Symposium d'Orsay sur la manipulation des symboles et l'utilisation d'APL Orsay (1973) vol. 2.

(2) BRAUER W.- Zur Bestimmung der maximalen Untergruppen des Transitionsmonoids eines Automaten.- Elektron. Informationsverarb..u. Kyb. 7 (1971) n° 4 251-160.

(3) BREED L.M. and LATHWELL R.H.- The Implementation of APL 360, in Interactive Systems for Experimental Applied Mathematics, M. KLERER and J. REINFELDS, Eds., Academic Press (1968) p. 390-399.

(4) CANNON J.J.- Computing the ideal structure of finite semigroups, Num. Mathematik 18 (1971) 254-266.

(5) CLIFFORD A.H. and PRESTON G.B.- The Algebraic Theory of Semigroups.- Vol. I. Amer. Math. Soc. (1961).

(6) COUSINEAU F.G., PERROT J.F. and RIFFLET J.M.- APL Programs for Direct Computation of a Finite Semigroup, in APL Congress' 73, North-Holland (1973) p. 67-74.

(7) DEMARS G. RAULT J.C. et RUGGIU G.- Le langage et les systèmes APL.- Paris, Masson (1974).

(8) EILENBERG S.- Automata, Languages and Machines.- Vol. A., Academic Press (1974)

(9) FALKOFF A.D. and IVERSON K.E.- The APL/360 Terminal System, in Interactive Systems for Experimental Applied Mathematics, M. KLERER and J. REINFELDS, Eds., Academic Press (1968), p. 22-37.

(10) GINZBURG A.- Algebraic Theory of Automata.- Academic Press (1968).

(11) HENNEMAN W.- The Automata Theorist's Helper.- Appendice à Mc NAUGTHON et PAPERT, Counter-Free Automata, M.I.T. Press (1971), p. 147-154.

(12) IVERSON K.E.- A Programming Language.- Wiley (1962).

(13) Mc NAUGTHON R. and PAPERT S.- The Syntactic Monoid of a Regular Event, in Algebraic Theory of Machines, Languages and semigroups, M.A. ARBIB, Ed., Academic Press (1968), p. 297-312; également dans Counter-Free Automata, des mêmes auteurs, M.I.T. Press (1971) p. 33-50.

(14) PERRIN D. et PERROT J.F.- Congruences et Automorphismes des automates finis.- Acta Informatica 1 (1971) 149-172.

(15) PERROT J.F.- Contribution à l'étude des monoïdes syntactiques et de certains groupes associés aux automates finis.- Thèse Sc. Math., Paris (1972).

(16) PERROT J.F.- Une théorie algébrique des automates finis monogènes.- Symposia Mathematica, Rome, 15 (1975) 201-244.

(17) ROBINET B.- Le langage APL.- Paris, Technip (1971).

J.F. PERROT
Institut de Programmation
11, Quai St-Bernard - PARIS V^{e}