RECHERCHES SUR LES SYSTEMES D'INFORMATION EN UTILISANT L'APL/LAVAL
(RESEARCH ON INFORMATION SYSTEMS USING APL/LAVAL)

Marion R. Finley, Jr.
Professeur agrégé
Département de Mathématiques - Université Laval, Québec

RESUME*

Des recherches en cours à l'Université Laval sur les systèmes d'information à très grande capacité sont décrites et le rôle joué par l'APL/LAVAL dans ces recherches est montré. Ce rôle de l'APL/LAVAL se voit surtout comme outil pour monter d'une façon rapide et convenable des projets-pilotes, ainsi donnant un feedback immédiat au développement théorique de la vérification expérimentale. Des propriétés saillantes du langage et du système APL sont esquissées et des suggestions au sujet d'une façon plus adéquate pour la manipulation des structures des données sont présentées. (Some of the current research at Laval University on large-scale information systems is described together with the role played in this research by the APL/LAVAL system. This role is seen above all as a means of rapidly and conveniently setting up pilot systems, thus yielding immediate feedback to theoretical development from experimental checking. Some of the outstanding features of both the APL language and system are sketched and suggestions for a more adequate way of manipulating data structures are outlined.)

(*) Travail appuyé par l'octroi no. 2651 du Ministère de l'Education du Québec et par IBM Canada.

1. LE DEFI

De toutes les sciences développées par l'homme, l'informatique serait peut-être la moins favorisée dans le sens que, après presque vingt-cinq années d'existence, il n'existe pas encore un langage convenable qui puisse exprimer les phénomènes principaux auxquels s'intéressent ses adhérents. Les sciences traditionnelles comme la physique et la chimie possèdent depuis assez longtemps des structures mathématiques et des notations qui sont à la fois pratiques, élégantes et compactes servant de charpente intellectuelle et conceptuelle à l'intérieur de laquelle les chercheurs puissent "penser" et imaginer leurs travaux expérimentaux et théoriques. Mais, en informatique, surtout dans ce qu'on peut appeler les sciences de l'information, il n'existe pas une façon cohérente de regarder l'univers de l'information et de son traitement. Au contraire, pour ne parler que des langages de programmation, selon Weinberg (1973) en citant Sammet (1967), il y a une vraie tour de Babel de langages- environ 117 développés entre 1952 et 1967, au taux moyen de vingt nouveaux langages par année depuis 1966. Ce qui est encore pire, c'est qu'il existe des systèmes qui dépassent la capacité de l'humain à les saisir, de sorte que l'usager n'est jamais complètement certain s'il aurait vraiment profité ou pas de la pleine puissance que lui offre le système.

Est-ce qu'il existe un langage universel, élégant et utile pour le traitement de l'information? La réponse à cette question à l'heure actuelle est certainement négative. Cependant, les désidérata d'un tel langage commencent à découler des recherche contemporaines en informatique, surtout de celles axées d'une façon ou de l'autre, sur les structures de données et leur traitement. Dans le présent article, le rôle de l'APL/LAVAL dans certaines recherches menées à l'Université Laval sur les systèmes d'information à grande capacité est montré et les propriétés saillantes du langage et du système APL sont esquissées afin de mettre en évidence la justification pour la prétension que l'APL représente une étape importante vers le développement d'un langage encore plus convenable pour la manipulation des structures d'information. Finalement, quelques réflexions sur le développement d'une façon plus cohérente de regarder les structures d'information sont présentées.

2. LE LANGAGE APL

Un programme (plus précisément, en terminologie APL, une fonction) en APL ressemble à un poème classique chinois: il est très compacte, ses symboles (opérations) ont une grande valeur sémantique, la syntaxe est simple, mais une ligne d'un programme peut avoir une signification assez compliquée et le programme est , en général, court, permettant ainsi une grande modularité. Le sigle "APL" est dérivé du titre du livre A Programming Language écrit par Dr. Kenneth Iverson en 1962. Le Dr. Iverson a proposé dans ce livre une notation quasi-algébrique permettant d'exprimer d'une façon compacte et cohérente la plupart des opérations rencontrées dans les

calculs effectués par l'ordinateur, surtout les opérations matricielles. Il propose quelques conventions intéressantes, à savoir:

(a) l'absence d'une hiérarchie imposée sur la séquence d'execution des opérations. Un énoncé est balayé de droit à gauche en effectuant les opérations dans l'ordre dans lequel elles sont rencontrées. Par exemple, l'énoncé $Z \leftarrow A + B \times C$ mettrait la valeur de $A+(B \times C)$ dans Z. Pour obtenir $(A+B)\times C$ on peut écrire, soit $(A+B)\times C$, soit $C \times A+B$.

(b) les opérations monadiques s'écrivent toujours à gauche de leurs opérandes et les opérations dyadiques entre ses deux opérandes. Cette même idée s'étend aussi aux fonctions (programmes ou sous-programmes) définies par le programmeur. Toutes les opérations simples sont représentées par des symboles simples et mnémoniques, e.g., la multiplication par $\times$, la division par $\div$, etc. Un grand nombre d'opérations qui sont souvent employées sont inclues dans le langage. En voici quelques-unes:

$\lceil X$	le plus petit entier $\geq X$
$\lfloor X$	le plus grand entier $\leq X$
$!X$	le factoriel de X
	etc.

(c) plus profondément, les opérandes ne sont pas déclarées d'avance. Leur type est défini plutôt dynamiquement, soit par une assignation, soit par l'entrée des données:

$X \leftarrow 1$; $Y \leftarrow 1.2349$; $Z \leftarrow$ 'UN NOM' (une chaîne)

$M \leftarrow \begin{pmatrix} 1 & 2 & 3 \\ 9 & 4 & 7 \end{pmatrix}$ (une matrice)

$L \leftarrow \begin{pmatrix} NOM1 \\ NOM2 \\ NOM3 \end{pmatrix}$ (une liste de noms en forme d'une matrice de lignes consistant en des chaînes de caractères.)

(d) les données peuvent être des matrices, des vecteurs, ou des scalaires. Les matrices sont de n'importe quelle dimension ($m \times n \times p \times q \times \ldots$) et, en effet, tous les types de données sont des formes matricielles si l'on considère que les vecteurs et scalaires ne sont que des cas spéciaux des matrices. Il y a des opérations spéciales pour la restructuration et le réarrangement des données. Une liste d'opérations intéressantes suit:

1-la linéarisation d'une matrice, c'est à dire, la tranformation d'une matrice $m \times n$ (i.e., un objet en deux dimensions) en un vecteur (objet en une dimension) ayant $m \times n$ éléments.

2-l'inverse de l'opération 1, i.e., la transformation d'un vecteur ayant $m \times n$ éléments en une matrice $m \times n$.

3- le tri des éléments d'un vecteur en ordre ascendant(ou descendant).

4- la conversion ou le codage d'un vecteur selon une base (donnée en forme de vecteur) et l'inverse, i.e., le décodage d'un nombre en ses composantes (le vecteur des produits des puissances de la base).

5- l'extension naturelle de toutes les opérations élémentaires aux matrices:

$\lceil X[1] \quad X[2] \quad X[3] \quad \ldots \quad X[N]$ = maximum des $X[I]$

$\lfloor X[1] \quad X[2] \quad X[3] \quad \ldots \quad X[N]$ = minimum des $X[I]$

$(X[1] \quad X[2] \quad \ldots \quad X[N]) \times (Y[1] \quad Y[2] \quad \ldots \quad Y[N])$

$= (X[1] \times Y[1] \quad X[2] \times Y[2] \quad \ldots \quad X[N] \times Y[N])$

$A \times B = C$, A,B,C $m \times n$, où C est le produit élément par élément de A et B.

6- des généralisations des produits internes et externes des matrices. Il existe aussi dans certaines versions implantées du langage une opération pour trouver l'inverse des matrices.
7- les chaînes des caractères sont considérées comme étant des vecteurs dont la longuer est égale au nombre de symboles contenus dans la chaîne.

Dans tous ces cas, les opérations sont représentées par des symboles simples et mnémoniques.

Le but de cette section n'est pas de présenter les détails de la programmation en APL. Pour ce faire, le lecteur peut consulter des ouvrages tels le volume de Gilman et Rose (1970). Pour montrer la puissance expressive du langage, quelques exemples sont développés à la Figure 1.

3. LE SYSTEME APL/LAVAL

Sans une implantation viable, aucun langage, quelle que soit sa puissance, ne peut être utile. Ce qui fait de l'APL un langage valable, ce sont ses versions opérationnelles implantées d'abord sur des ordinateurs IBM et ensuite sur les machines d'autres manufacturiers tels CDC et XEROX. Une machine APL a été proposée par Abrams (1970) et même un mini-ordinateur APL a été construit (MCM-70).

En voici quelques caractéristiques saillantes, communes à la plupart des systèmes APL (pour plus de détails, voir les références données à la fin de l'article):

(a) la plupart des systèmes APL sont des interprètes qui opèrent dans un environnement à temps partagé. Il y a, avec quelques exceptions seulement, très peu de compilation des programmes en APL: chaque fois qu'un énoncé sera exécuté, il devrait être décodé par l'interprète. Il y a, cependant, certaines versions qui gardent les fonctions(programmes) en forme partiellement décodée.

(b) il y a un ensemble de "commandes de système" qui permettent à l'usager de manipuler des segments de mémoire appelés "blocs de travail", de les stocker dans les "bibliothèques"("libraries") publiques ou privées, de les envoyer en mémoire active, de poser des questions au système et à d'autres usagers, etc.

(c) il existe un "éditeur" pour assouplir la création, la correction et la modification des fonctions. Une fonction, ainsi que les blocs de travail, peut être "barré" pour empêcher les personnes non-authorisées de les regarder ou de les modifier.

(d) la plupart des versions disponibles jouissent d'un système de fichiers (Galbraith, 1973 , et I.P. Sharpe).

(e) la procédure par laquelle l'usager entre en communication avec le système ou rompt contact avec le système est assez simple. Il est très difficile d'endommager le système.

(f) le niveau de sécurité des "bibliothèques" et des fichiers est raisonnable.

Figure 1. Exemples de la Programmation en APL

(a) Exemple d'un tri

L'énoncé ? 15 ⍴ 25 engendre un vecteur de longueur 15 dont les composantes sont des entiers choisis au hasard entre 1 et 25. La variable X prend ce vecteur comme valeur et la boîte □ sort le vecteur sur la console. Ensuite l'opération ⍋X (le "grade-up" de X) trie les composantes de X et donne comme valeur le vecteur des indices originaux des composantes arrangés de sorte que celles-ci seront ordonnées en ordre ascendant:

⍋X = 13 6 14 1 5 ...,

i.e., X[13] = 1 est la première composante en ordre ascendant
X[6] = 2 " " deuxième " " " "
X[14] = 2 " " troisième " " " ",

et ainsi de suite. Donc, X[⍋X] donne le vecteur dont les composantes sont celles de X triées en ordre ascendant.

Ensuite, 50 ⍴ '.' engendre un vecteur de longueur 50 de caractères, ici le point, et le met dans la variable TRAIT, ensuite la boîte "quote" l'imprime.

(b) L'adressage "associatif"("content addressing")

La notation APL permet des fois d'exprimer certaines opérations qui sont logiquement equivalentes à des opérations parallèles, telles l'adressage associatif. Ici LISTE est une matrice de caractères de 10 lignes par 10 colonnes dont les lignes sont des chaînes. L'opération

'ASIMOV ' ∧.= ⍉LISTE

donne le vecteur V d'indices où V[I]=1 si le nom ASIMOV se trouve à la I-ième ligne et 0 autrement. L'opération V/⍳10 donne les indices I où V[I]=1 et LISTE [V/⍳10;] sort la (ou les) ligne(s) dont V[I]=1.

(c) Tri d'une liste de noms

L'opération ALPH⍳⍉LISTE crée une matrice où les colonnes sont des vecteurs des nombres des positions dans l'alphabet ALPH des lettres du nom correspondant, e.g. ASIMOV devient col(1,19,9,13,15,22). Ces vecteurs sont codés à la base 27 (le nombre d'éléments de ALPH) ainsi donnant un vecteur de valeurs numériques, soit les noms codés. Ce vecteur W est trié et l'énoncé LISTE [⍋W;] donne finalement la liste des noms de LISTE arrangée en rodre alphabétique.

(d) Une fonction intéressante

Cet exemple montre une fonction appelée CLASSE qui ne contient que trois énoncés et aucune boucle qui effectue, cependant, une opération assez compliquée, soit la suivante: étant donnée n'importe quelle chaîne de mots (séparés par des blancs), sortir la liste de tous les noms ayany lieu das la chaîne sans répétition et en indiquant le nombre de répétitions. Trop compliqué à expliquer ici, cet exemple met en évidence la concision et la puissance du langage. L'exemple est donné grace à M Michel Pilote, étudiant gradué en Mathématiques (option informatique) qui l'a trouvé en réponse à un problème posé dans Hellerman, Digital Computer System Principles, McGraw-Hill, 1973, pp 86-87.

Figure 1 (suite)

```
      ⎕←X←?15⍴25
 4  19  12  14   6   2  17  17  24  10  13  21   1   2  14      (a)

      ⍋X
13   6  14   1   5  10   3  11   4  15   7   8   2  12   9

      X[⍋X]
 1   2   2   4   6  10  12  13  14  14  17  17  19  21  24

      ⎕←TRAIT←50⍴'.'
..................................................

      LISTE                                                    (b)
EINSTEIN
ARCHIMEDE
ARISTOTE
LAPLACE
PLATON
POINCARRE
WAGNER
MOZART
ASIMOV
NEWTON
      ⍴LISTE
10  10
      'ASIMOV    '∧.=⍉LISTE
0  0  0  0  0  0  0  0  1  0

      ('PLATON   '∧.=⍉LISTE)/⍳10
5
      LISTE[5;]
PLATON
      TRAIT
..................................................

      ALPH                                                     (c)
 ABCDEFGHIJKLMNOPQRSTUVWXYZ

      LISTE[⍋27⊥ALPH⍳⍉LISTE;]
ARCHIMEDE
ARISTOTE
ASIMOV
EINSTEIN
LAPLACE
MOZART
NEWTON
PLATON
POINCARRE
WAGNER
```

Figure 1 (suite)- (d)

```
      ⍝ ***** PROGRAMME    *****

      ∇CLASSE[⎕]∇
    ∇ CLASSE S
[1]   M←(⍴ALPH)⊥ALPH⍳⍞L←(S,' ')[⍞B⌊(⍴B←⍞MRK∘.+⍳10)⍴1+(MRK←((1+K)≠1+K,1+⍴S)/K←(S=' ')/⍳⍴S),1+⍴S←' ',S]
[2]   CR,'LISTE ET REPETITIONS          LISTE ORDONNEE',CR
[3]   MASQUE\LIST,(⍞'123456789*'[10⌊+/⍞L×.=⍞LIST]),(LIST←L[⍋L;])[⍋M⌈M←((⍳⍴M)≠⍳M)/⍳⍴M];]
    ∇

      ⍝ ***** VARIABLES    *****

      ALPH
 ABCDEFGHIJKLMNOPQRSTUVWXYZ
      MASQUE
0  0  0  1  1  1  1  1  1  1  1  1  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
      1  1  1  1  1  1  1  1  1  1
      CHAINE
LES AMIS DES AMIS DES AMIS DES AMIS DE MES AMIS SONT MES AMIS ET TOUJOURS ILS LE SERONT
      CR←'B'

      ⍝ *****  EXEMPLE     *****

      CLASSE CHAINE

LISTE ET REPETITIONS          LISTE ORDONNEE

   LES          1                AMIS
   AMIS         6                DE
   DES          3                DES
   DE           1                ET
   MES          2                ILS
   SONT         1                LE
   ET           1                LES
   TOUJOURS     1                MES
   ILS          1                SERONT
   LE           1                SONT
   SERONT       1                TOUJOURS
```

L'APL/LAVAL est une version de l'APL implantée à l'Université Laval, d'abord sur l'IBM 360/50 et ensuite sur l'IBM 370/145 munies du système de fichiers de I.P. Sharpe. Plusieurs améliorations et innovations ont été apportées au système, telles les fonctions "globuls", la fonction " " (execute), l'opération pour l'inversion des matrices et d'autres dont la description est hors les limites du présent article. Pour plus de renseignements, voir Samson et Robichaud(1972), Miville des Chênes et Robichaud (1972), et Raynard, Robichaud et Simian(1972). Le système APL/LAVAL peut desservir environ 65 usagers à la fois, avec une moyenne d'environ 35. Le temps de réponse reste assez bon, même lorsque le système est aux limites de ses capacités. L'accès au système peut bien être son aspect le plus charmant , surtout vue la grande fiabilité du système.

Pour une esquisse de la structure globale de l'APL/LAVAL, voir à la Figure 2.

4. DESCRIPTION DU PROJET "ETUDES DE SYSTEMES A TRES GRANDE CAPACITE"

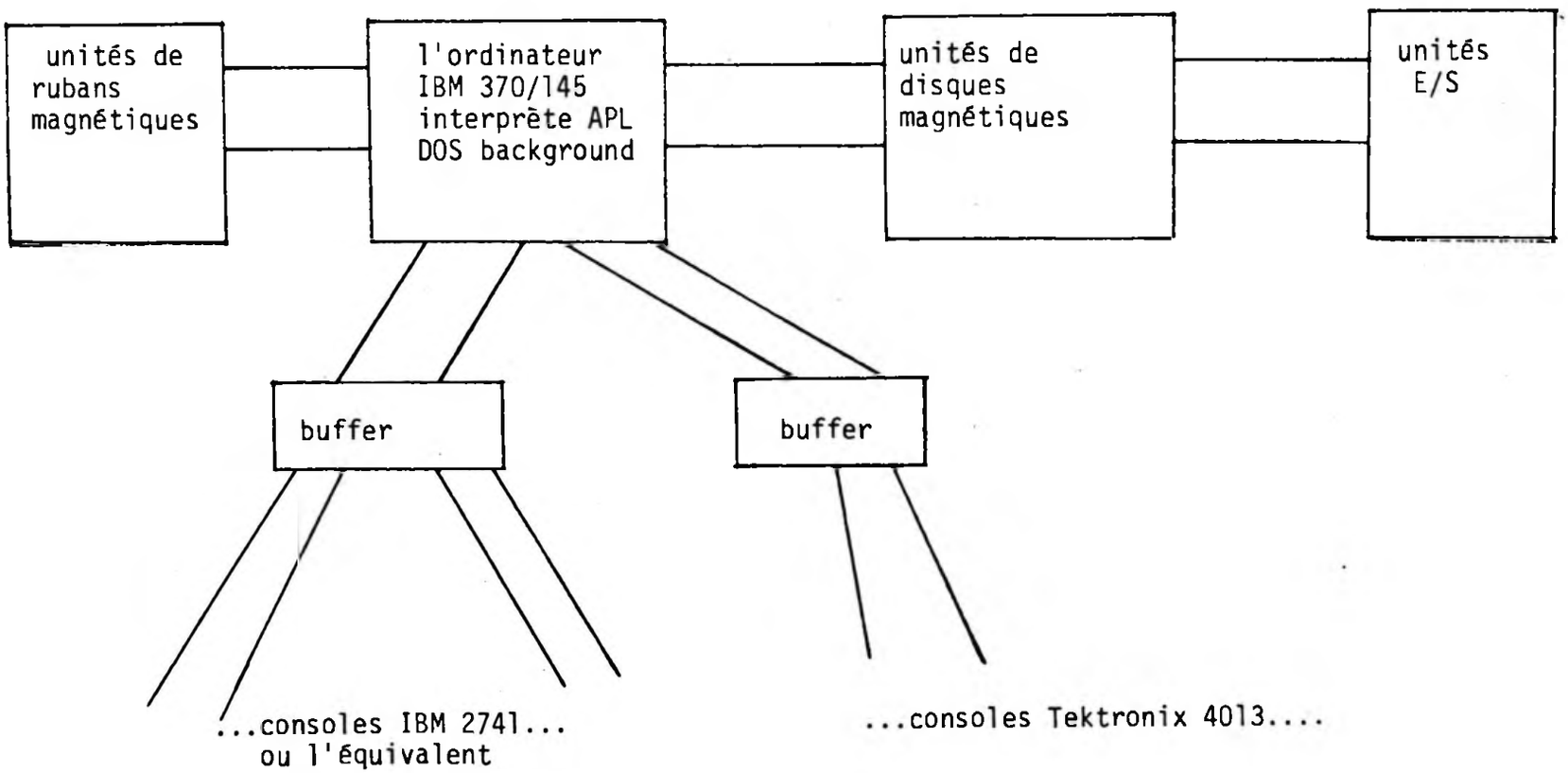
Depuis trois ans, une équipe de chercheurs à l'Université Laval s'est penchée sur l'étude de systèmes d'information à très grande capacité. Le projet comporte deux aspects qui se recouvrent partiellement, l'un axé sur la classification des données, l'autre sur le traitement par ordinateur des langues naturelles. Une courte description de ces deus aspects suit:

(a) la classification. D'un intérêt particulier est le cas où la banque de données comporte des documents dans des domaines comme le droit, les sciences descriptives, le management, et ainsi de suite, où l'information requise sera souvent décrite en langue naturelle. Du travail a été effectué sur le développement d'une théorie générale de la classification (Hatcher et Coray, 1974), et ensuite sur le stockage et le dépistage (Khadem et Giroux, 1973) avec des applications. En général, on a cherché des techniques et des méthodes de classification des sujets d'un domaine donné. L'idée directrice est toujours d'organiser ces sujets afin qu'ils puissent être chargés en mémoire et retrouvés efficacement. On espère par la suite trouver une théorie mathématique pouvant servir de base à des techniques plus évoluées. Les résultats obtenus à date permettent d'entrevoir la possibilité de trouver une telle théorie. Voici une liste de quelques problèmes traités:

1-le développement de modèles pour la classification et l'indexation de l'information.
2-la structuration de l'information en mémoire interne et externe et l'interface entre les structures employées et les modèles proposés en 1.
3-les langages de dialogue homme-machine permettant à l'humain de dialoguer avec un système d'infromation afin d'y retrouver ou d'y insérer de l'information.
4-une considération du côté du hardware et la recherche des appareils les plus appropriés aux exigences soulevées dans 1 à 3.

Les modèles trouvés sont implantés en version proto-type opérationnelle

Figure 2. Structure globale du système APL/LAVAL



pour étudier en plus de profondeur leurs propriétés pratiques et pour réviser l'approche théorqiue , si nécessaire. Parmi ces modèles on peut mentionner:

-des systèmes MIS(management information systems)
-des systèmes EAO(enseignement assisté par ordinateur)
-des dictionnaires interactifs de langues naturelles (chinois, arabe, et français).

Pour plus de détails, voir Finley et Schmidt (1974), Bourassa et Finley(1973), et Fournier(1972).

(b) langues naturelles. L'intérêt ici est de développer des modèles des langues naturelles permettant à l'usager de sommuniquer plus facilement avec une banque de données. et de faire en même temps de la traduction d'une langue source an langue-cible. Pour l'instant, l'attention a été dirigée vers des modèles comportant une analyse syntaxique et une analyse sémantique convenables, puis vers la structures des dictionnaires exigées par ces deux types d'analyse. Le problème de l'entrée et de la sorite des documents en des langues ayant des alphabets exotiques,soit l'arabe et le chinois, a été considéré(Finley, 1973). La langue chinoise avec son écriture si difficile pose de serieux problèmes et a reçu une attention particulière impliquant des retombées dans l'affichage graphique et les structures des données en général.

5. L'APL/LAVAL COMME OUTIL DE RECHERCHE

L'APL/LAVAL a joué un rôle primordial dans des recherches achevées en tant que véhicule primaire pour l'étude des systèmes proto-types. Les raisons viennent des avantages offerts par l'APL, soient

-l'accès facile au système
-le système de fichiers
-la modularité intrinsèque du langage
-la concision et la puissance du langage
-la possibilité de traiter assez efficacement des objets non-numériques
-la fiabilité et la sécurité du système.

Ces avantages permettent au chercheur de construire des systèmes-pilotes de taille moyenne sans trop d'ennuis. En effet, le design d'un système-pilote se fait assez souvent en temps réel simultanément avec la programmation. Donc, les idées développées sont susceptibles d'une vérification expérimentale directe.

A cause du feedback rapide donné par le système APL, la procédure de l'écriture et de la correction de porgrammes est raccourcie et simplifiée. Ceci a beaucoup stimulé les lignes de pensées des recherches menées et a amené à son tour à des recherches sur l'APL lui-même d'une part, sur le problème général de la description des structures de données d'autre part.

Le développement rapide de la technologie du hardware ouvre la possibilté que l'APL sera encore plus efficace dans l'avenir en tant qu'outil de recherche sur les grandes banques de données. D'abord, le traitement des fichiers deviendra plus souple, permettant un accès facile à de très grands fichiers. Deuxièment, le langage sera sans doute assoupli à l'égard du traitement des objets

non-matriciels, comme les listes ou les arbres (voir Brown, 1971). Troisièment, les consoles visuelles et le software pour l'affichage graphique deviendront plus commodes et moins dispendieux (voir Bork).

En effet, même sans ces développements, l'APL est loin d'être un "langage jouet" comme l'implique Weunberg (1971). Il nous offre plutôt des possibilités intéressantes , même en ce qui concerne les applications concrètes dans le monde des affaires (Hurtubise et Poulin, 1974).

6. STRUCTURES DE L'INFORMATION

La commodité du système APL soulève la possibilité de créer un système encore plus approprié pour la manipulation des structures d'information. Ce système peut être basé sur l'APL ou au moins bâti dans l'esprit de l'APL. Le grand désavantage de l'APL est le suivant: ses structures naturelles de données sont des matrices et toutes ses opérations puissantes sont axées sur la manipulation de ce type d'objet. Donc, il n'y a pas de façon directe de créer et de manipuler des objets tels les listes, les réseaux de noeuds, les arbres, etc. Il est à noter, cependant, que Iverson avait proposé des structures arborescentes dans son livre (Iverson, 1962).

Quelques idées au sujet de la formalisation du concept de structures de données qui découlent des recherches faites jusqu'à date sont esquissées ci-dessous. Elles représentent les premiers pas pris vers un langage plus idéal du traitement de l'information.

1-une structure ("objet") est définie indépendamment du support physique
2-un objet peut être n'importe quoi-une liste, une chaîne, un programme, un objet consistant en d'autres objets, etc.
3-il y a un ensemble d'objets primitifs au moyen desquels tout autre objet peut être construit.
4-les objets peuvent porter des étiquettes ou ils peuvent être étiquettés par une partie d'eux-mêmes. Les étiquettes peuvent être données expressément ou implicitement (telles les lignes d'une matrice).
5-les objets peuvent exister en une relation hiérarchique l'un avec l'autre. Pour éviter des contradictions menant à des culs-de-sacs ("deadlocks") dans le fonctionnement du système, un type ou niveau logique sera affecté aux objets. Ces contradictions ressemblent aux paradoxes de la logique mathématique.
6-il existe des opérations pour la configuration des objets.
7-toutes les opérations standards sont inclues, telles l'insertion, l'extraction, l'enlèvement,le réarrangement, et la recherche d'un objet donné.

Pour des exemples, voir à la Figure 3.

Figure 3. Idées sur les structures de données

On tente de formaliser le concept de structure de donnée en se servant des relations et de systèmes de relations. Par une relation on entend un soue-ensemble du produit logique d'ensembles donnés (pas nécessairem -ent distinct). On donne à ces relations des interprétations (signifi-cations sémantiques et pragmatiques). On peut se servir de ces relations pour en construire d'autres relations plus complexes. Un objet sera alors l'un des objets primitifs ou la valeur d'une relation . Quelques exemples devraient éclaircir ces idées, qui ont trouvé des applications à l'affichage graphique des caractères chinois (Finley, 1974).

LISTE (a,b) : arranger les objets a et b en liste- a,b

L ← LISTE(a,b) définie L comme la liste a,b

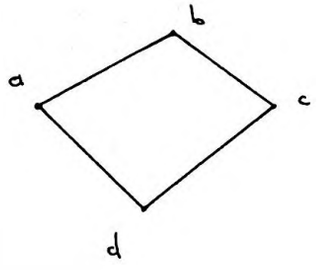
L ← LISTE(L,c) remplace L par la nouvelle liste a,b,c

L ← LISTE(L,L) crée la liste L(1) de listes L(2)- ici, quoique la nouvelle liste ait le même nom que les autres, son niveau est 1, celui des autres 2.

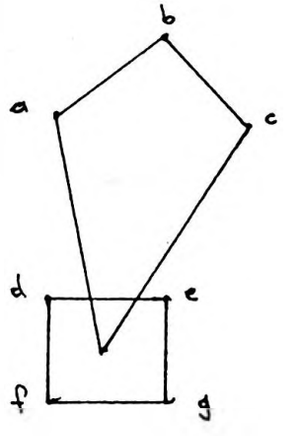
QUAD(a,b,c,d) : tracer un segment (du centre de masse) de a à b
" " " b " c
" " " c " d
" " " d " a
" " "

où a,b ,c, d sont données en coördonées relatives. Donc,

QUAD(a,b,c,d) peut donner la forme suivante:



tandis que QUAD(a,b,c,QUAD(d,e,f,g)) pourrait donner



7. CONCLUSIONS

Le langage et le système APL se sont montré comme constituant une voie d'accès facile à l'ordinateur et la puissance de ce dernier. A l'intérieur de ses limitations, la langage APL a une très grande souplesse en ce qui concerne la manipulation des données. Le principal point faible c'est le fait que les structures de données les plus naturelles sont les matrices (de dimension quelconque). Cependant, des structures plus générales peuvent être traitées, ce qui a permis le développement de systèmes-pilotes intéressants.

Il y a toujours un certain incovénient à cause du besoin de faire l'interprétation des énoncés.

Le concept de l'APL, soit de construire un langage-système concis et puissant, permettant la manipulation d'objets générailsés, a directement stimulé certaines recherches pouvant à mener à des améliorations à l'APL lui-même ou à un nouveau langage.

Donc, l'APL n'a pas été qu'un véhicule convenable pour la mise en marche de systèmes-pilotes, mais il a aussi été une source d'idées pertinentes au problème de la description et de la manipulation des structures de données.

REFERENCES BIBLIOGRAPHIQUES

ABRAMS, P.S. An APL Machine, T.R. No.3, D.S.L.' Stanford University, 1970.

BORK, A.W. "APL as a language for interactive graphics", plus de renseignements disponibles sur cet article.

BOURASSA, M. et M.R. FINLEY, rapport en préparation, 1974.

BROWN, J.A. A Generalization of APL, PhD Dissertation, Syracuse University,1971.

FINLEY, M.R. "On the Formal Description of Chinese Characters"' First International Symposium on Computers and Chinese I/O Systems, Academia Sinica, Taipei, 1973.

FOURNIER, S.F. "Enseignement assisté par ordinateur", rapport interne, Département de Mathématiques, Université Laval, 1972.

GALBRAITH, D.S. DREV APL Under UTS, DREV Inf 13/72, April 1972.

" The DREV APL File System, DREV M-2261, 1973.

GILMAN, L. et A.J.ROSE APL/360 An Interactive Approach, Wiley, 1970.

HATCHER, W.S. et CORAY, "A Logical Framework for Large File Information Handling", à paraître dans Information Sciences, 1974.

HURTUBISE, R. et Y. POULIN "Coûts informatiques d'un SIG appuyé sur un système APL", E.N.A.P., Université du Québec, Québec, 1974.

IVERSON, K.E. A Programming Language, John Wiley, 1962.

KHADEM, R. et P. GIROUX, "Dépistage de l'information pour de larges fichiers", Information Storage and Retrieval October 1973.

MIVILLE DES CHENES, A. et L.P.A. ROBICHAUD Subtasking in APL, CTI/LAVAL, 1972.

RAYNARD, Y., ROBICHAUD, L.P.A., et G.SIMIAN Description et simulation de systèmes informatiques à l'aide du langage APL, CTI/LAVAL, 1972.

SAMSON, D. et L.P.A.ROBICHAUD APL/360, CTI/LAVAL, 1972.

SCMIDT, M. et M.R.FINLEY , rapport en préparation, 1974.

WEINBERG, G.M. The Psychology of Computer Programming, van Nostrand, 1971.