& APPLICATIONS DE MARSEILLE
L. I. T. A. M.
DEPARTEMENT DE MATHEMATIQUE-INFORMATIQUE
LUMINY
UNIVERSITÉ AIX-MARSEILLE II

ISSN 0291 - 5413

Sommaire

Juin 1984

ADRESSE POSTALE : FACULTÉ DES SCIENCES DE LUMINY
MATHÉMATIQUE-INFORMATIQUE - LITAM - BAT. TPR2 - 9ème ETAGE
CASE 901 - 70, ROUTE LÉON LACHAMP - 13288 MARSEILLE CEDEX 9 - ☎ (91) 41.01.40 POSTE 3294 - 3110

Editorial

e. bianco

L'informatique et le néant.

Pour le Larousse,l'éditorial est un article qui exprime les vues de la direction d'un journal.Notre bulletin n'est pas un journal,mais il est dirigé vers l'informatique et de plus,il a une vue sur l'avenir.Aussi dans notre éditorial essayons-nous d'analyser l'actualité afin d'entrevoir comment s'y insère l'informatique.A partir de là,plus ou moins involontairement on est amenés à faire des pronostics sur les devenirs de l'enfant.

Comment résister au plaisir de se dire: j'ai deviné ce qui va se passer dans vingt ans,de l'écrire,afin qu'au bout du délai on n'ait plus qu'à se retourner en disant: "C'était écrit ... par moi." .

Oui,mais si on se trompe ? Ma foi rien n'oblige à exhumer les vieux cadavres.

Comment se caractérise l'actualité.Pas grand'chose de bien neuf ne me semble caractériser l'actualité.La technique avance bien un peu,désormais,les expressions: "Etablir la communication" "Faire toute la lumière","Eclairer le monde" sont en train de devenir étroitement synonymes avec la popularisation de la fibre optique.

Bien.

Mais quoi d'autre de tellement important ? Ah oui,l'Europe. J'oubliais mais j'ai quelques excuses.Ce n'est pas la première fois qu'on nous fait le coup.Combien de dictateurs combien de capitaines rêvaient d'être les maîtres de cette oeuvre incertaine.

L'un des derniers en date,le dictateur Charlot,s'est même fait péter le globe dans la figure.Et depuis,Rocard n'a pas été soutenu par ses serfs.L'affaire baigne dans un brouillard laiteux.

Que reste-t-il de l'actualité ? Quand,par chance,on gravite autour d'une élection,on se délecte à coup de pronostics tous plus informatiques les uns que les autres.Et pour ne pas trop gâcher ces admirables joutes de plus-un-pour-cent par ci et de moins-deux-points par là,on suspend tout dans un grand silence angoissé,huit jours avant l'acte.

Et quand,d'un coup,notre prestidigitateur soulève le chapeau: à l'intérieur,horreur,le serpent a avalé quelques grenouilles.Ca change un peu le citoyen qui lui,dans son ordinaire,avale plutot des couleuvres.

Mais il ne faudrait tout de même pas oublier la crise.Si l'on se reporte trente ans en arrière,vers les années cinquante, c'étaient les soucoupes volantes qui faisaient frémir les foules depuis,il a fallu les peindre en ovni pour soutenir l'intérêt, malgré cela elles ont bien pâli.

Alors on abreuve les foules de crise.Là,au moins,la menace est bien réelle: chômage et gaspillage sont les deux mamelles d'une bonne gestion.

Qui dit bonne gestion dit informatique,c'est bien connu.Et je m'inquiète sur le devenir de l'informatique Européènne.Pourtant

il suffit que je regarde par exemple Eurotechnique pour me rassurer.C'était une maison de verre,(au moins pour une bonne moitié),depuis peu elle a pris l'opacité d'une solide maison bien de chez nous.

D'autres indices sont tout aussi éloquents,notre ordinateur universitaire,le Micral,passe de mains en mains comme un objet brûlant; en ce moment Bull semble tenir le bon bout,d'ailleurs son long périple à travers les Etats-Unis l'a certainement mûri.

Un dernier point enfin pour attiser notre enthousiasme: les crédits de recherche paraissent miraculeusement avoir disparu.Ce ne saurait être que pour accroître notre potentiel national,et je suis fier de penser que cet argent va enfin servir à acheter des brevets Américains ou Japonais.La recherche fondamentale représentait vraiment,chez nous,un gaspillage innomable,alors que les étrangers font ça tellement mieux.

J'ai vécu la dernière guerre,et je me souviens parfaitement du rêve que caressait Monsieur Hitler: l'accouplement de la France agricole avec l'Allemagne industrielle.Il suffirait de peu de choses en plus pour faire une bonne Europe: par exemple une Angleterre s'occupant des assurances maritimes,d'une Suisse chargée des logiciels informatiques importants,la France pourrait à la rigueur faire des casseroles aluminium à titre de promotion exceptionnelle,quant aux réfrigérateurs,plus délicats,l'Italie conviendrait parfaitement.

On peut d'ores et déja constater que le système d'heure changeante deux fois l'an,inauguré par le Troisième Reich,a soigneusement été remis en place ...

Après la recherche fondamentale,il est naturel de s'intéresser

à la culture tout court.Un bon pas a été fait dans ce sens puisque notre ministre d'icelle a interdit la vente des livres à bon marché.Nous sommes en bonne voie.

Logiciellistes de tous poils à vos plumes,cocorico au bec,du match de foot-ball synthétique au carnet de la ménagère en cassettes,l'avenir de l'informatique est à votre portée au même titre que le chômage.

SUR UN RESULTAT DE SHANNON A PROPOS DES MACHINES DE TURING A DEUX ETATS

(J.Ph.Lehmann)

Subject classification informatics: F1.1

Résumé: Toute machine de Turing à m symboles et n états peut être ramenée à une machine de Turing à 3mn+2m symboles et 2 états, et ce de façon que:

- Le procédé de codage qui fait passer de la configuration initiale de la machine à simuler à la configuration de la machine réduite à 2 états, soit l'identité sauf en un point,
- à la fin de chaque phase de calcul, qui dans la machine réduite à 2 états, simule une phase élémentaire de la machine initiale, le code de la configuration est encore obtenu par l'identité sauf en un point,
- pendant ces phases de calcul intermédiaires, le code de la configuration est obtenu par l'identité sauf en deux points au plus.

A INTRODUCTION

I. RAPPELS SUR LES RESULTATS DE SHANNON

I1) LE RESULTAT ESSENTIEL

Dans un article de 1956, publié dans Automata studies, Shannon établit le résultat suivant:
Soit une machine de Turing, dont l'alphabet comprend m symboles et possédant n états. On peut alors construire une machine de Turing équivalente à la précédente qui possède 2 états et dont l'alphabet comprend 4mn+m symboles

I2) AUTRES RESULTATS

a) Le résultat précédent pouvant s'appliquer à une machine de Turing quelconque, il vaut en particulier pour le cas d'une machine universelle, et donc il est possible de construire une telle machine à 2 états..

b) Il est montré également qu'il est impossible de construire une machine universelle de Turing à un seul état.

c) Egalement qu'à toute machine de Turing à m symboles et n états, on peut associer une machine équivalente n'utilisant que 2 symboles et au plus 8mn états.

I3) HYPOTHESES ET REMARQUES

L'article se termine sur quelques remarques indiquant qu'il est possible dans une certaine mesure d'échanger symboles et états, et l'hypothèse est formulée que ceci peut se faire, sans trop changer le produit nombre d'états nombre de symboles.

II. REMARQUES

Le résultat énoncé en I2c) est bien connu; il est appuyé sur le fait qu'il suffit(et que c'est nécéssaire) de 2 signes pour coder n'importe quel alphabet.

Le résultat énoncé en I2b) forme la référence initiale de travaux que nous avons en cours et dont une partie est déjà disponible. Notre attention sera ici essentiellement dirigée vers les points I1) etI3):

- D'une part nous proposons une amélioration de la majoration du nombre des symboles de la machine à 2 états, tout en respectant les contraintes déjà signalées sur le codage des configurations.

- D'autre part, à la vue de ce résultat, on peut d'ores et déjà indiquer que la vraisemblance des hypothèses formulées au I3) ne parait pas s'être renforcée.

B LES ELEMENTS DE LA METHODE DE SHANNON

Nous ne nous préoccupons plus ici, que de la construction, à partir d'une machine quelconque, d'une machine équivalente à 2 états.
Bien qu'il soit difficile, pour ne pas dire impossible de résumer la démonstration de Shannon autrement qu'en l'exposant, nous donnons un aperçu général de la méthode:

A est la machine initiale, B la machine à 2 états qui va simuler B.

Puisque B ne dispose que de 2 états, ceux-ci ne peuvent servir comme substituts des états de A; il faut donc opérer par voie logicielle en représentant les couples symbole-état, pour chaque phase du calcul opéré par A.

Ceci ne peut se faire qu'au prix d'un accroissement de la taille de l'alphabet initial.

Vont donc apparaître dans l'alphabet de B, à l'entrée de chaque phase de calcul, des symboles indiquant à la fois, quels étaient le symbole lu et l'état, à cet instant dans la machine A.

Le problème consiste alors à opérer la transformation de forme requise, et à faire en sorte qu'on se retrouve en un temps fini, devant la case appropriée (celle de gauche ou celle de droite), et que celle-ci contienne à son tour un symbole qui indiquera à la fois, le nouvel état et le nouveau symbole qui sont associés à la phase de calcul suivante.

Pour ce faire, Shannon imagine un processus de rebond entre la case considérée à l'entrée du calcul et celle qu'il faut ensuite atteindre; au cours de ce processus, le symbole écrit à chaque rebond joue le rôle d'un compteur qu'on contrôle jusqu'à son débordement, en même temps qu'il continue d'indiquer quel était le symbole de A, qu'il représentait, au début de la phase de calcul.

Shannon explique qu'il faut donc disposer, outre les m symboles qui désigneront les symboles de la machine A, de 4mn autres indiquant: les couples état-symbole (mn)

si la case du ruban considérée,reçoit ou transmet de l'information dans le processus de rebond(X2),
si cette même case commande un déplacement à droite ou bien à gauche (X2);
Au total il faut ainsi 4mn+m symboles.

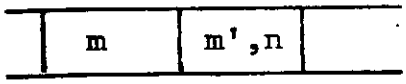
C REDUCTION DE L'ALPHABET

L'argument général qu'on vient d'exposer repose sur le caractère indispensable des renseignements décrits plus haut, et en effet, on ne voit pas comment il serait possible de se passer de ces indications; de ce fait l'alphabet semble être réduit au minimum. C'est cette dernière conclusion qui est inexacte: en imposant au seul alphabet de véhiculer les informations en cause, on se prive des possibilités offertes par l'existence de 2 états; c'est en tirant parti de leur existence, et notamment en leur faisant jouer des rôles (quasiment) symétriques, qu'on va pouvoir améliorer la majoration initiale.

Auparavant, pour que la méthode soit clairement exposée, nous étudierons un problème voisin, débarrassé des détails techniques secondaires, et qui permet de mieux comprendre l'essentiel du procédé.

I UN PROBLEME VOISIN

D



F

Les deux configurations présentées ci-dessus, représentent l'état du ruban au début d'une phase de calcul (D), puis à la fin de celle-ci (F). Il faut passer d'une configuration à l'autre, et la machine n'a que deux états.

En d'autres termes, disposant d'une information complexe(m,n) située dans une case du ruban, comment procéder pour maintenir une partie de celle-ci (m), transporter une autre partie (n), le tout sans détruire l'information présente dans la case vers laquelle on effectue le transport.
Le lien avec le problème qui nous occupe est clair:(m,n) représentera un couple état-symbole; à la fin du calcul, on aura réussi à faire transiter l'état n dans la case contiguë, sans perdre celle qui s'y trouvait déjà, et en conservant le symbole m dans la case initiale.
Bien entendu, le problème qui nous est effectivement posé est soumis à d'autres contraintes: l'information m devra eventuellement être changée, l'état à transporter ne sera pas forcément n, enfin le déplacement à opérer n'a aucune raison d'être à droite; on verra malgré tout, que ces points ne sont pas essentiels.
Précisons enfin quelques points de notation: certains symboles sont désignés par des couples de lettres; il n'y a là aucune distinction qui opposerait des doubles symboles à des symboles simples: il est commode de choisir des noms qui nous indiquent le sens qu'il faut attacher aux choses ainsi nommées.
Par exemple si m varie dans un ensemble à 3 éléments, et n dans un ensemble à 2 éléments, nous aurions à décrire les situations:

m = 1		1
m = 2		2
m = 3		3
m,n = 1,1		4
m,n = 1,2	qu'il suffirait de coder ainsi	5
m,n = 2,1		6
m,n = 2,2		7
m,n = 3,1		8
m,n = 3,2		9

Passer alors de la configuration:

3,1	2

à la configuration:

3	2,1

signifierait passer de :

8	2

à :

3	6

Ceci précisé, nous utiliserons dans ce qui suivra la notation à double lettre (et même plus tard à triple lettre), qui permet de mieux suivre le déroulement des opérations.

Nous allons résoudre ce premier problème à travers un exemple; on se convaincra sans peine qu'il n'y a là aucune perte de généralité: supposons que nous voulions passer d'une situation qui serait celle-ci

5,3	9	D

à celle-là

5	9,3	F

Nous définissons le schème de Turing suivant:

	I	II
5,3	5,2/DII	
9		9,1/GI
5,2	5,1/DII	
9,1		9,2/GI
5,1	5,0/DII	
9,2		9,3/GI
5,0	5 /DI	
9,3	et le processus reprend...	

Comme on peut le constater, le procédé consiste à se servir de l'information initiale comme d'un compteur qu'on décrémente;

tant qu'il n'atteint pas 0, on incrémente la case voisine; dans ces conditions, le nombre des soustractions est égal à celui des additions, et le transfert se réalise comme on le souhaitait, et les informations 5 et 9 ne sont pas détruites dans le calcul.

II LE PROBLEME GENERAL

Retournant au problème initial, il faut réaliser certaines adaptations:

- Au moment où on entre dans une phase de calcul, nous ne laisserons pas intacte, la partie de l'information représentative du symbole lu, mais nous la remplacerons par le symbole déterminé dans le schème de Turing de la machine A.
- Quant à l'autre partie de l'information, représentative de l'état dans le schème de A, ce n'est pas elle qu'on fera transiter vers la case contiguë convenable, mais celle représentant l'état déterminé dans le schème de A.
- Enfin il nous restera à gérer le déplacement, dont le sens nous est également connu à partir du schème de A.

Le codage dans le ruban de B est exactement le même que pour celui de A; les doubles-lettres n'interviennent que durant les phases de calcul où l'on réalise le transfert d'information qu'on a décrit plus haut
Ainsi, dans B, une phase de calcul qui simule une phase élémentaire de A, se caractérise par le fait qu'il y a dans 2 cases contiguës du ruban, présence d'une double lettre;

la fin d'une phase se caractérisera par le fait qu'il n'y a de double-lettre que dans une seule case.
Il faut initialiser le processus en marquant dans la case pointée par l'index, la double lettre représentative du signe lu et de l'état à l'instant initial dans A.

Là encore nous exposerons les choses à travers un exemple, ce qui permettra au lecteur de beaucoup mieux suivre les calculs. Au fur à mesure nous essaierons de dégager les arguments qui montrent que nos choix sont quasi obligés; en d'autres termes, au lieu de définir a priori, un alphabet convenable, nous le construisons progréssivement comme les nécéssités le déterminent:

Supposons donc, qu'à l'entrée d'une phase de calcul, nous lisions l'information 3,7; cette information a été constituée par le déroulement antérieur du processus, et au départ par les conditions initiales qui ont été explicitées plus haut.
Supposons d'autre part, que dans A le schème indique:

$$3,q_7 \longrightarrow 5,q_4\ D$$

et que dans la case située à droite de celle qu'on examine dans B, il y ait 8.

Parmi les deux états nous en distinguons un, noté I que nous appellerons état d'entrée de phase; ce sera toujours le même à chaque entrée, puisqu'à ces instants précis, il n'y a absolumment rien à mémoriser des calculs passés, si ce n'est justement le fait qu'ils sont terminés, et que donc on entre dans une nouvelle phase.

3,7	5,4,X/DII
8	8,1/GI
5,4,X	5,3,X/DII
8,1	8,2/GI
5,3,X	5,2,X/DII
8,2	8,3/GI
5,2,X	5,1,X/DII
8,3	8,4/GI
5,1,X	5/DI
8,4	Le processus peut reprendre....

3,7	8	
I		
5,4,X	8	
	II	
5,4,X	8,1	
I		
5,3,X	8,1	
	II	
5,3,X	8,2	
I		
5,2,X	8,2	
	II	
5,2,X	8,3	
I		
5,1,X	8,3	
	II	
5,1,X	8,4	
I		
5	8,4	
	I	

Un examen attentif montre que l'inscription de 5,4,X au début du processus est inévitable: on ne saurait se passer de constituer l'information imposée par le schème ainsi que l'indication de l'état dans lequel il faudra traiter la case à atteindre; il faut alors pouvoir distinguer les informations état-symbole, qui jouent le rôle de compteur de celles qu'on trouve à l'entrée de phase. C'est pourquoi il faut étendre l'alphabet par les mn signes i,j,X.
On voit alors qu'il n'est pas nécéssaire d'indicer davantage les symboles lus dans l'état II.

Il reste qu'on pourrait imaginer de commencer le calcul en demeurant dans l'état I; la suite montrera qu'on aboutirait à une solution symétrique de celle qu'on expose.

D'ores et déjà on peut indiquer une partie de la structure du schème de la machine B:
Si dans A nous avons n états et m symboles,
Alors pour toutes les règles de A qui sont de la forme:

$$i,q_j \longrightarrow i'/D\ q_{j'}$$

Nous définissons les règles de B suivantes:

$$(i,j),I \longrightarrow (i',j',X)/DII$$

$$i,II \longrightarrow (i,1)/GI$$

$$(i,j,X),I \longrightarrow (i,j-1,X)/DII \qquad j=2,..,n$$

$$(i,j),II \longrightarrow (i,j+1)/GI \qquad j=1,..,n-1$$

$$(i,1,X),I \longrightarrow i\ /\ DI$$

Il faut donc disposer, pour l'instant, de 2mn+m symboles:
La première règle fait apparaître 2mn symboles et la seconde m autres. Les règles suivantes ne font appel à aucun symbole nouveau.
Il reste à traiter le cas où le déplacement imposé serait G; nous reprendrons l'exemple déjà considéré, en supposant que dans le schème de A il y ait la règle:

$$3,q_7 \longrightarrow 5,q_4\ G$$

et que dans la case située à gauche, il y ait 8.

	I	II
3,7	5,4,X/GI	
8	8,1,Y/DII	
5,4,X		5,3,X/GI
8,1,Y	8,2,Y/DII	
5,3,X		5,2,X/GI
8,2,Y	8,3,Y/DII	
5,2,X		5,1,X/GI
8,3,Y	8,4,Y/DII	
5,1,X		5,0,X/GII
8,4,Y		8,4/DI
5,0,X	5/GI	
8,4	Le processus peut reprendre.....	

8	3,7
	I
8	5,4,X
I	
8,1,Y	5,4,X
	II
8,1,Y	5,3,X
I	
8,2,Y	5,3,X
	II
8,2,Y	5,2,X
I	
8,3,Y	5,2,X
	II
8,3,Y	5,1,X
I	
8,4,Y	5,1,X
	II
8,4,Y	5,0,X
II	
8,4	5,0,X
	I
8,4	5
I	

Seule une observation attentive permet de saisir les choix qui ont été opérés; on peut néammoins donner les indications suivantes: au départ on conserve le même état, ainsi le symbole 8 pourra être traité différemment du cas précédent. En ne faisant pas intervenir des symboles d'un type nouveau à l'entrée du calcul, on réalise l'essentiel de l'économie.

On notera que les symboles du type i,j,X sont réutilisés mais que,ils ne sont plus vus que dans l'état II.
La création de symboles du type i,j,Y est indispensable, car dans l'état I, les formes i,j,X définissent une décrémentation (cas du mouvement à droite) alors qu'ici le même état determinera une incrémentation, et les formes i,j sont associées à une entrée de phase.
D'autre part on observera que les formes i,j,X ont du être étendues au cas i,0,X; il n'est pas possible en effet, lorsque le symbole i,1,X est découvert dans l'état II, de passer directement à la forme i comme on l'avait fait dans le cas précédent, car sinon la redécouverte de la forme i, que ce soit dans l'état I ou l'état II, est interprétée comme un deuxième pas de calcul (2ième ligne des tableaux ci-dessus).
Pour résumer, la clef consiste à faire jouer, dans le cas du déplacement à gauche, aux états I et II, les rôles qui étaient tenus respectivement par les états II et I, lors du déplacement à droite.

Nous pouvons maintenant donner le reste des règles de la machine B:
Pour toutes les règles de A qui sont de la forme:

$$i,q_j \longrightarrow i'/G\ q_{j'}$$

nous définissons les règles de B suivantes:

$$(i,j),I \longrightarrow (i',j',X)/GI$$

$$i,I \longrightarrow (i,1,Y)/DII$$

$$(i,j,X),II \longrightarrow (i,j-1,X)/GI \quad j=2,\ldots,n$$

$$(i,j,Y),I \longrightarrow (i,j+1,Y)/DII \quad j=1,..,n-1$$

$(i,1,X),II \longrightarrow (i,0,X)/GII$

$(i,j,Y),II \longrightarrow (i,j)/DI$

$(i,0,X),I \longrightarrow i\ /GI$

Il a fallu créer les symboles notés i,j,Y. Ceux-ci sont au nombre de mn. D'autre part, nous avons du étendre lensemble des formes i,j,X, aux cas i,0,X. Ceux-ci sont au nombre de m.

Au total l'alphabet ainsi determiné comprend:

2mn+m (déplacement D)

mn+m (déplacement G)

soit 3mn+2m symboles; la majoration initiale (4mn+m) est donc améliorée.

BIBLIOGRAPHIE:

Shannon: A universal Turing machine with two internal states. Automata studies. 1956.

UNE DEFINITION DU CALCULABLE

(J.Ph.Lehmann)

Subject classification informatics: F1.1 F4.1

Résumé: on présente un type de machine qui a la puissance du calculable.

-A-

INTRODUCTION

Dans cette théorie, il faut examiner avec soin le statut des symboles. Ils ne seront pas ici, considérés comme des abstractions, mais comme des corps physiques concrets. La question qui se pose alors, est celle de la description précise du protocole qui permet de les découvrir, les isoler, les reconnaitre et les distinguer.

Nous ne développerons pas ici, car le temps nous fait défaut, une théorie du symbolisme qui devrait se situer,d'une part,à la fois sur le plan physique et sur le plan mathématique, d'autre part, dans un cadre méthodologique voisin de ce qu'on appelle phénoménologie, dans le vocabulaire philosophique. Pourtant, à l'observateur attentif, il n'échappera pas que ces préoccupations sont présentes dans ce qui va suivre.

Celles-ci sont déjà visibles dans la théorie de Turing: un équipement spécial permet de parcourir les chaines de symboles et les isoler; la façon même dont le déplacement est défini, annonce, ce qui, pour le bloc de traitement (B.T.),est un symbole; de sorte que, la

division du ruban en cases, doit être vue, non pas en soi, mais en liaison avec les performances du bloc de traitement. Il n'y a qu'une commodité à séparer le ruban et le B.T., et si l'on souhaite être précis, il faut considérer leur couple et ne pas omettre ce fait: c'est la taille du mouvement du B.T. qui définit la taille de la case.

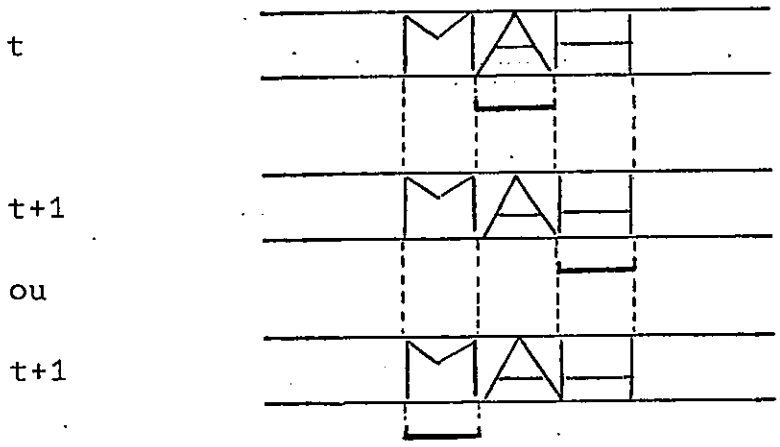
-B-

EXPOSE

I DESCRIPTION DE LA MACHINE

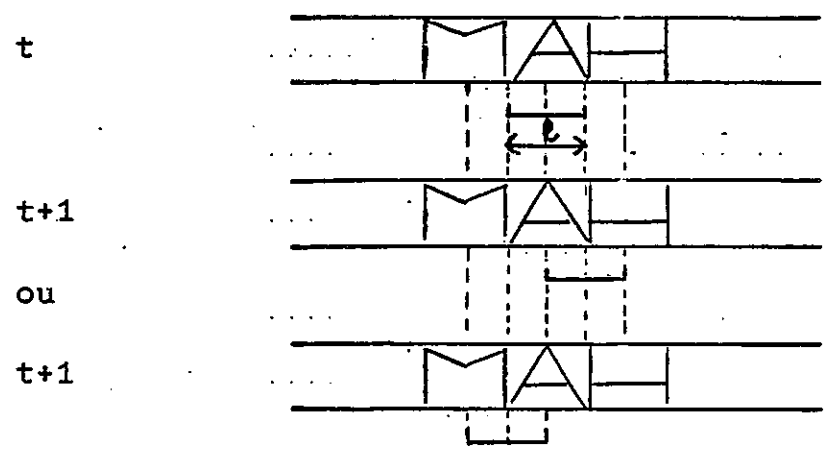
Comme on vient de l'indiquer, c'est le mouvement qui définit l'entité élémentaire reconnue par la tête (B.T.) et traitée par elle.

Dans la machine de Turing (M.T.), les symboles tous normalisés, occupent l'espace d'une case dont la taille est L; le mouvement peut alors se représenter ainsi: +L ou -L.

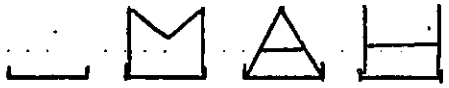


La machine que nous définissons, est munie, elle aussi d'un B.T., dont on verra les caractéristiques plus

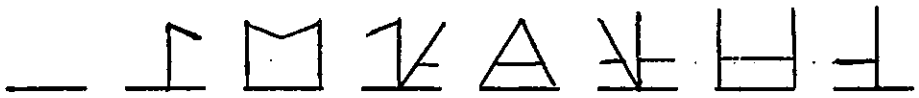
loin; le champ qui lui est associé est de Taille L, mais le mouvement est de taille $\frac{L}{2}$.



Alors que dans la M.T., les symboles apparaissant au B.T. étaient:



ici ils peuvent être:



Bien entendu, les symboles de l'alphabet latin, utilisés ci-dessus, ne sont là que pour servir une image intuitive; d'autre part nous ne voyons aucune raison d'écarter de telles images, quand leur puissance évoca-

trice est indéniable.

II DEUX NIVEAUX LINGUISTIQUES

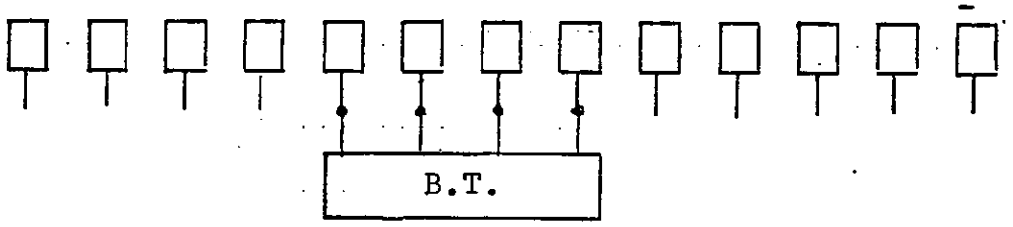
- Dans la théorie de Turing, certaines choses sont passées sous silence et notamment celle-ci: les symboles sont tantôt considérés par le B.T., tantôt par un observateur extérieur qui interprète la totalité du système. Bien que ces deux types d'observation ne constituent en rien le même phénomène, il n'y a alors aucun gain à les distinguer, et c'est ainsi que l'entendement courant, inconsciemment les confond.

- Dans la présente théorie, il n'en va pas de même:

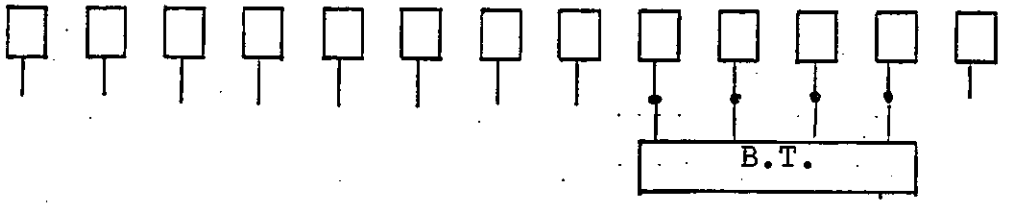
Du point de vue de l'observateur, il est souhaitable d'exprimer la succession des situations observées, de façon à mettre en évidence que la forme soumise à un instant donné au B.T., présente certaines caractéristiques communes avec celle qui a été produite au pas précédent. En d'autres termes, les symboles successivement traités, sont, du point de vue de l'observateur dans des rapports de forme précis.

Du point de vue du B.T., ceci ne peut être pris en considération.

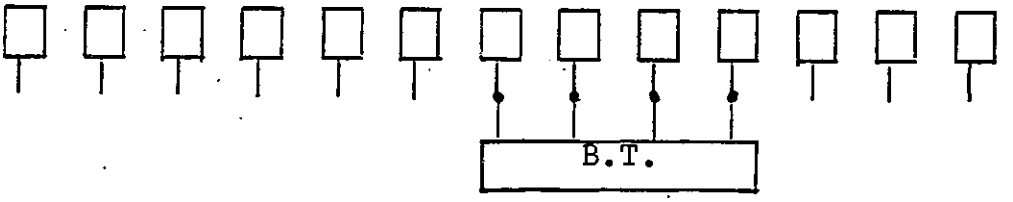
Pour éclaircir ces points imaginons un codage binaire, $2^n \geqslant k$, k taille de l'alphabet à coder, et un B.T. qui parcourt des mots de taille n; dans le dessin ci-dessous n=4:



Dans la M.T., un déplacement à droite serait celui-ci:



alors qu'ici nous aurons:



Les petits carrés représentent des cellules à 1 bit. Il est clair que dans les deux types de machines le B.T. reçoit à chaque instant une parmi les 2^n configurations possibles, et qu'aucune autre indication ne lui parvient. Si nous avions choisi l'octet comme taille du mot pour

le B.T., nous utiliserions alors pour l'observateur, une désignation hexadécimale, qui aurait l'avantage d'exprimer la suite des actions de façon très claire. Découvrant par exemple 111100100111

noté F 2

et opérant un mouvement à droite (sans changement des symboles vus), on découvrirait alors

00100111

noté 2 7

-C-

RESULTATS

I : LA MACHINE AINSI DEFINIE A LA PUISSANCE DU CALCULABLE

I1) DESCRIPTION DANS CE LANGAGE D'UNE M.T.

Dans ce qui suivra les déplacements de la M.T. seront notés G ou D, ceux de notre machine g ou d.
Une M.T. est définie par une configuration initiale,

$$(a_{i_1}, a_{i_2}, \ldots\ldots\ldots, a_{i_p})$$

$$a_{i_j} \in \{a_1, a_2, \ldots\ldots, a_m\} \qquad j=1,2,\ldots,p$$

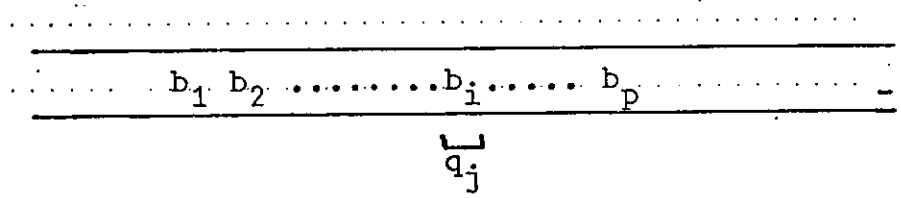
un ensemble fini de règles de la forme,

$$a_i, q_j \longrightarrow a_{i'}, q_{j'}, X \qquad X=D \text{ ou } G$$

une condition d'initialisation qui indique le symbole lu au départ, et l'état dans lequel celui-ci est vu.

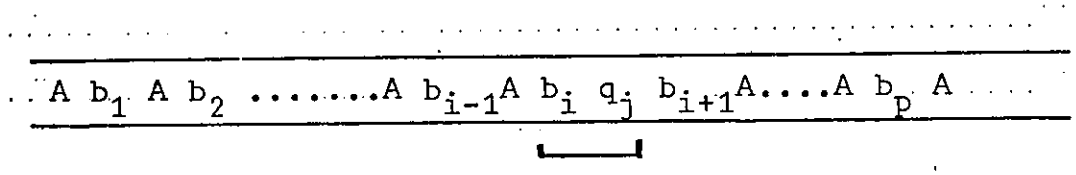
Il nous faut en premier lieu définir dans notre système un codage associé à la configuration initiale qui a été définie dans la M.T.. En fait nous le ferons pour la configuration courante, et il sera clair, par induction et par regression, que le processus se developpe correctement.
Supposons qu'à un instant donné la configuration cou-

rante de la M.T. soit:



$b_i \in \{a_1, a_2, \ldots a_m\}$

La configuration associée de notre machine sera:



1er cas: si la règle de Turing applicable est:

$$b_i, q_j \longrightarrow b_i', q_j', D$$

nous définissons les règles suivantes:

$$b_i q_j \longrightarrow b_i', q_j', d$$

$$q_j', b_{i+1} \longrightarrow A b_{i+1}^{j'}, d$$

$$b_{i+1}^{j'} A \longrightarrow b_{i+1} q_j^{+}, d$$

$$q_j^{+}, b_{i+2} \longrightarrow q_j', b_{i+2}, g$$

Les symboles ainsi définis pour le B.T. sont alors au nombre de $((n+1)(m+1)+n)^2$ si m est le nombre des symboles de la M.T. et n celui des états. Pour l'instant aucun effort spécial de minimisation n'a été fait.

Pour le cas que nous venons de décrire, la suite des configurations est celle-ci:

$$A \quad \underbrace{b_i \quad q_j} \quad b_{i+1} \quad A \quad b_{i+2}$$

$$A \quad b_{i'} \quad \underbrace{q_{j'} \quad b_{i+1}} \quad A \quad b_{i+2}$$

$$A \quad b_{i'} \quad A \quad \underbrace{b_{i+1}^{j'} \quad A} \quad b_{i+2}$$

$$A \quad b_{i'} \quad A \quad b_{i+1} \quad \underbrace{q_j^{+'} \quad b_{i+2}}$$

$$A \quad b_{i'} \quad A \quad \underbrace{b_{i+1} \quad q_{j'}} \quad b_{i+2}$$

2ème cas: si la règle de Turing applicable est:

$$b_i, q_j \longrightarrow b_{i'}, q_{j'}, G$$

nous définissons les règles suivantes:

$$b_i q_j \longrightarrow b_i^{j'} A \ , \ g$$

$$A b_i^{j'} \longrightarrow q_j, b_{i'} \ , \ g$$

On notera qu'il n'y a pas de nouveaux symboles à définir, car les situations précédentes ne peuvent jamais

se confondre avec celles décrites juste avant.

Pour le cas que nous venons de voir, la suite des configurations est celle-ci:

$$A \; b_{i-1} \; A \; b_i \; q_j \; b_{i+1}$$

$$A \; b_{i-1} \; A \; b_{i'}^{j'} \; A \; b_{i+1}$$

$$A \; b_{i-1} \; q_{j'} \; b_{i'} \; A \; b_{i+1}$$

I2)DESCRIPTION EN LANGAGE M.T. DE CES MACHINES

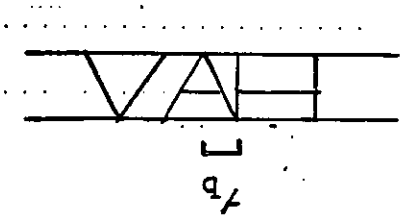
Nous traitons la question sur un exemple; supposons que la configuration courante de notre machine soit celle-ci:

et que la règle applicable soit:

$$a_i a_j \longrightarrow a_{i'} a_{j'} , \; d$$

nous devons aboutir à la situation:

Nous considérons alors une M.T. ainsi définie, avec la configuration associée:



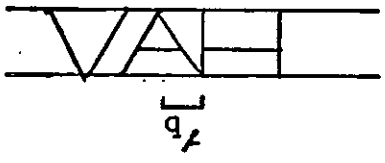
et on définit les règles:

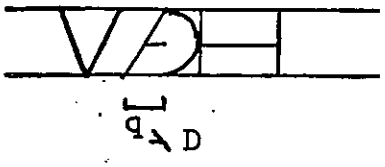
$\lambda, q_f \longrightarrow \supset, q_{\lambda D}$, G

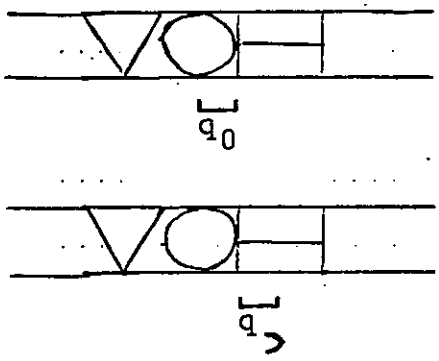
$f, q_{\lambda D} \longrightarrow c, q_0$, D

$\supset, q_0 \longrightarrow \supset, q_{\supset}$, D

Dans ces conditions la suite des configurations sera:







Il reste à examiner le cas où la règle applicable est de la forme:

$$a_i a_j \longrightarrow a_I a_j \ , \ g$$

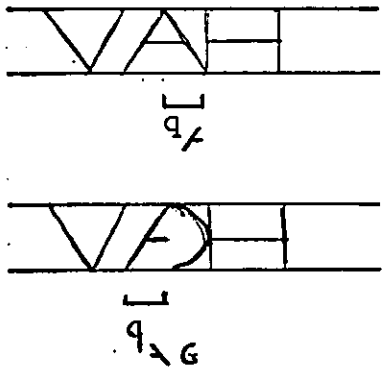
Il suffit alors de définir les règles:

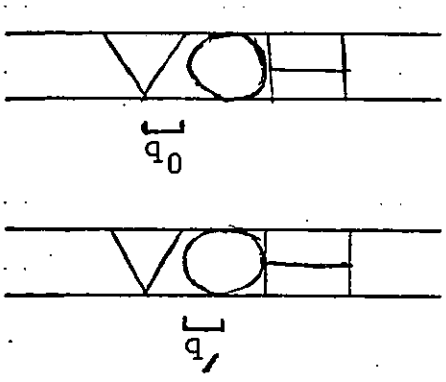
$$\lambda, q_f \longrightarrow \supset, q_{\lambda G}, \ G$$

$$f, q_{\lambda G} \longrightarrow C, q_0 \ , \ G$$

$$/, q_0 \longrightarrow /, q_/ \ , \ D$$

L'évolution des configurations est alors celle-ci:





De façon générale, si la machine initiale est munie d'un alphabet à m symboles, on détermine par le procédé qu'on a exposé 2m symboles pour la M.T. associée; Le nombre de ces états sera 2m (pour tous les q_{a_i})

+ 2m (pour tous les $q_{a_i\ D}$)

+ 2m (pour tous les $q_{a_i\ G}$)

+ 1 (pour q_0)

soit 6m+1.

II LES MACHINES AINSI DEFINIES N'ONT QU'UN SEUL ETAT

Ce simple énoncé qui découle immédiatement de ce qui précède, est porteur de nombreuses conséquences, la première d'entre elles étant que la notion d'état, n'est pas indispensable à une définition cohérente du calculable,

en ce sens qu'il est possible de concevoir le B.T., comme réduit à une simple fonction booleenne.

- D -

DEVELOPPEMENTS

I DIRECTIONS

Une fois l'équivalence avec les autres formes du calculable établie, il est possible de développer cette théorie de façon propre : pour que le modèle proposé ait une certaine autonomie, il est nécéssaire que ce développement puisse se faire à partir du terrain intuitif sans qu'il soit forcé de recourir aux autres définitions. Actuellement nous travaillons dans ce sens et il nous est possible d'annoncer:

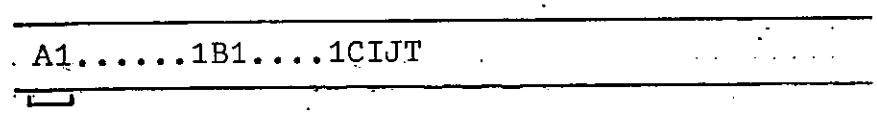
- La construction d'une machine universelle sans changement d'état, où le codage sera très simple, et le recours à d'autres définitions du calculable, inutile.

- Plusieurs raffinements:

notamment au niveau du mouvement: si on exige dans cette théorie, que les symboles puissent en quelque sorte être découpés, nous montrerons qu'il n'y a là aucune restriction vraiment forte; plus précisément nous travaillons à donner de ce calcul plusieurs autres modèles, en particulier un à décalage du B.T. minimum, et un autre à décalage maximum.

II UN EXEMPLE

Pour conclure ces notes, nous donnons un exemple de calcul dans ce modèle, la multiplication:



Entre A et B d'une part et B et C d'autre part, il y a deux nombres, et leur produit va être déposé entre I et J. Seules les règles qui mènent à une modification du symbole observé sont signalées; si le déplacement n'est pas indiqué, il est sous-entendu se faire à droite.

$A1,AA^S$	JT,J^-T	jusqu'à ce que:
$A^S1,A1$	$1J^-;1^-J$ g	$1J,11$
$B1,BB^S$	$I1^-,I^-1$ g	" "
$B^S1,B1$	CI^-,C^-I g	B^SC,B^-C g
$IJ,I1$	$1C^-,1^-C$ g	BB^-,B^-1 g
$1T,1J$	$11^-,1^-1$ g	jusqu'à ce que:
	$B1^-,BB^S$	$1B^-,1^-B$ g
		$A1^-,AA^S$ g

et enfin A^SB qui conclut la phase de calcul.

Autojection et Compilateurs

Procédure Formelle Symbolique

e. bianco

C.R. Subject Classification Informatics 4.12 4.21 4.22

Résumé.

Cet article apporte à la Procédure Formelle Symbolique un complément de structures destiné à prendre en compte les notions du fini illimité. La notion de file support est destinée à rédiger commodément au niveau du système le passage d'information entre programmes disjoints dans leur écriture et leur déroulement.

AUTOJECTION ET SYSTEMES

PROCEDURE FORMELLE SYMBOLIQUE.

Dans le premier chapitre de l'article,j'ai défini la partie de la procédure formelle symbolique qui recouvre le champ du fini borné.Il me faut alors construire la partie de ce langage destinée à décrire le système,c'est-àdire ce qui concerne le fini illimité,et aussi tout ce qui va servir au raccord entre fini borné et fini illimité.

Bien entendu,quelles que soient les propriétés dont on peut munir un langage,elles ne seront jamais suffisantes pour résoudre tous les problèmes qui se posent à cette occasion.Le cadre dans lequel je me place,parcontre: l'autojection,m'apporte les structures extérieures au langage qui me sont nécéssaires.

A l'exception d'un seul,tous les algorithmes manipulés à l'aide de la PFS sont autojectifs.Celui qui ne l'est pas est précisément le système.Cet algorithme n'est rien d'autre en effet qu'une boucle indéfiniment répétitive,et distribuant des quanta de travail aux diverses tâches en présence.

Je vais concevoir le langage pour qu'il me facilite le travail dans deux sortes de circonstances.L'une qui a trait à l'existence des tâches,avec mise en oeuvre des échanges,introduction et sauvegarde des programmes et des données.L'autre dans laquel-

le se manifestent les rapports des tâches entre-elles.

La prise encompte des échanges fera l'objet d'un exposé ultérieur.Je vais me préoccuper des rapports qui existent entre les tâches,et entre les tâches et le système.Ces rapports ne sont pas simples à exprimer avec un langage de programmation, il s'agit de phénomènes complèxes,et le point de vue choisi peut lui-même influer sur cette complèxité,en général en l'aggravant.Je m'en tiendrai donc à la méthodologie définie plus haut que je considère comme une bonne approche du phénomène.

Prenons l'exemple simple d'un code produit par un certain programme,disons un traducteur,ce code doit être repris par un autre programme,disons un dérouleur.Traducteur et dérouleur sont construits séparément et ils travaillent également de façon séparée.Donc leurs variables sont indépendantes,j'entend par là que chaque programme ignore les variables de l'autre.Le produit du premier,très généralement,appartient au fini illimité. Il faut donc prendre des précautions pour que le second reconnaisse l'information dont il a besoin.

Ce sont ces précautions que je vais condenser dans quelques notions utiles pour la procédure formelle symbolique.

Comme on l'a vu au chapitre précédent,les structures de files constituent des cadres bornés pour l'étalement des données et la commodité de manipulation des éléments provient de l'utilisation du statut.C'est dans certaines de ces files que sont construits les produits à transmettre.Or,rarement le produit occupera exactement la file déclarée.Il faut donc pouvoir,soit condenser,soit fournir des extensions.

C'est pour contrôler le résultat de ces opérations que je dé-

finis le statut dynamique,dont le code permet de transmettre l'information de structure.

J'introduis donc un type de file supplémentaire indispensable pour le développement des mécanismes du système.La file support joue ce rôle qui consiste essentiellement à assurer la compatibilité entre des calculs disjoints.C'est dans cette file que se transmettent les valeurs du statut dynamique.Les calculs qui concernent donc ce type de statut sont rattachés directement à la file support.

J'introduis également deux commodités supplémentaires dans la manipulation des tableaux et des files dynamiques.La file messages,destinée à contenir des données auto-interprétables, et un mode de calcul des indices d'élément dans une ligne de tableau.

FILE SUPPORT.

Cette file se définit sur une suite d'éléments minimum.Sa structure doit comporter une souplesse suffisante pour supporter les autres types de files.Le statut joue dans ce cadre un rôle particulier: pour chacun des éléments repérés par les références,il mesure la dimension en nombre d'éléments.Ce statut est calculable,sa valeur est un entier,et elle représente ce nombre d'éléments.L'entier précisé entre-parenthèses définit le statut de l'élément composant.Cette file a une structure de séquence au sens du MIL.

exemple.

support 10 000 , 100 , x(2) : y1 : y2 ;

représente une file de 10 000 éléments-minimum,de 100 élé-

ments ou lignes,et dont les statuts de lignes seront représentés par la variable "x".Le statut de l'élément composant a pour valeur: 2.Ce pourrait être le cas d'une mémoire à mots de 16 bits,si l'élément-minimum était l'octet.

On applique sur cette structure des instructions de la forme:

```
y1 := initial ;
x  := 15 ;
```

qui créent dans cet exemple un premier élément dont le statut précise qu'il comporte 15 composants de 2 éléments-minimum chacun.

La différence de structure entre file dynamique et file support apparaît clairement: dans l'une,la ligne est définie par un statut choisi dans une liste déclarée avec la file dynamique.Le statut sert à construire une ligne qui est donc constituée d'une suite de composants,chacun d'eux étant un agglomérat d'éléments minimum.Dans l'autre,la ligne se compose d'un seul élément qui se subdivise en composants.Le composant est défini de façon unique dans la file support,comme un agglomérat d'éléments-minimum.

De plus,la file support est conçue pour que la ligne puisse être subdivisée autrement en tenant compte du statut dynamique, dont la valeur elle-même est prise dans la file.Il devient ainsi nécessaire de se donner une sorte de méta-statut standard utile pour la récupération de la valeur du code du statut dynamique. Après cela un calcul permet d'atteindre les autres éléments.

La file support a la particularité de se composer d'éléments ou de lignes qu'on peut manipuler de deux façons: globalement avec un statut calculable attaché à la file,et localement avec un statut dynamique dont la valeur est à prendre dans l'élément lui-même.

CONSTRUCTION DE LA FILE SUPPORT.

La file support va servir au système pour réaliser un découpage de la mémoire afin d'en assurer la gestion.Il faut donc se donner les moyens de calculer les références de chacun des éléments qui composent la file.Ce travail est spécifique du support, je me donne une instruction spéciale.Reprenons l'exemple précédent:

support 10 000 , 100 , x(2) : y1 : y2 ;

supposons qu'on veuille découper la mémoire en deux parties de 3 000 et 7 000 éléments-minimum,j'écris:

```
x := 1 500 ;
y1 := initial ;
ligne y1 , x ;
x := 3 500 ;
y1 := → y1 ;
ligne y1 , x ;
```

L'instruction ligne note le couple référence-statut qui conserve cette valeur jusqu'à ce qu'une nouvelle instruction ligne la modifie.

La file étant partiellement ou complètement chargée,il peut se révéler nécessaire de retrouver la valeur du statut d'une ligne déterminée.Je donnerai une instruction spécifique de ce genre de traitement.

FILE MESSAGE.

Les tableaux et les files dynamiques sont des structures commodes pour traiter des codes qui sont définis directement du point de vue algorithmique.Mais il existe toujours une information dont les codes ne peuvent être reconnus directement de ce point de vue. C'est,par exemple,le cas d'une liste,mettons d'entiers,à longueur

paramétrée: il faut noter en mémoire le nombre de ces entiers,et c'est la valeur de ce nombre qui conditionne la reconnaissance des entiers.

Je me donne donc un type de file dans lequel le statut est calculé.Du point de vue structure,la file message est proche de la file support,mais elle se place dans le champ fini borné.

Exemple:

```
message 15 , 1 , (h,1) : m1 : m2 ;
                h := 1 ;
               y1 := initial;
              [y1,h := "G" ;
                h := h + 1 ;
              [y1,h := "L" ;
                h := h + 1 ;
              [y1,h := "U" ;
```

Cette file message comporte 15 éléments minimum,et une seule ligne."h" est un indice qui correspond à une correction d'adresse.La déclaration de "h" n'est pas indispensable.

STATUT DYNAMIQUE.

Le statut dynamique est un statut qui doit être transmis entre deux programmes disjoints à la fois dans leur écriture et dans leur déroulement.Bien sûr ils ne peuvent être disjoints dans leur conception.

Le statut dynamique est représenté par son code dans la ligne de la file support.Pour la cohérence du langage je prends à titre de convention un méta-statut destiné à:

1) permettre de retrouver le code du statut dynamique à la lecture de la ligne,
2) permettre d'explorer le code lui-même,pour reconstituer le

statut local de la ligne.

Pour le premier point,je décide de placer le code statut systématiquement comme premier élément de la ligne.Or,le statut de ce code étant de toute évidence variable,c'est le cas prévu en 2),il me faut des moyens spéciaux pour le manipuler.

Exemple:

```
support 100 K , 10 , h(1) : x1 : x2 : xm ;
                x1 := initial ;
                x1 := 3 → x1 ;
                [x1 := statut (3) , (2) , (2) , (5) ;
```

Ceci signifie que dans la ligne N°4 de cette file support j'écris un statut qui définit quatre composants de 3 , 2 , 2 , 5 , éléments minimum.En fait la ligne N°4 sera découpée en cinq éléments,le premier comporte comme valeur,celle du statut que la dernière instruction met en place.

Pour lire le statut ainsi mis en place en tête d'une ligne de file support,il me faut d'une part une structure capable d'accueillir une valeur dont j'ignore le statut a priori,et d'autre part une instruction spécifique afin de laisser ce statut implicite.

Deux indications peuvent être précieuses pour calculer,la valeur elle-même qu'il doit être possible de récupérer,afin de travailler dessus et la longueur occupée par le code.

Exemple:

```
          1 ( long(2) , ent2(2) ) : s1 : s2 ;
message 1 000 , 3 , (h,1) : n: t ;
                n := initial ;
                s1:= initial ;
          [s1,long := longstat [x1 ;
          [n := statut [x1 ;
```

Les deux opérateurs statut et longstat permettent de récuperer dans des éléments de files la valeur du statut dynamique et le nombre de cases que son code occupe.

à suivre