THÈSE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le titre de

DOCTEUR INGENIEUR

(TRAITEMENT DE L'INFORMATION)

par

Laurent MOUSSU

MODELE FONCTIONNEL DE MACHINES ASSOCIATIVES APPLICATIONS AU TRAITEMENT DE LA PAROLE

Thèse soutenue le 18 février 1981 devant la Commission d'Examen

Membres du Jury	Président	C. CARREZ
	Rapporteur	V. CORDONNIER
	Examinateurs	J.C. GENTINA
		J.C. POREE

P R O F E S S E U R S 1ERE C L A S S E

M. BACCHUS Pierre	Mathématiques
M. BEAUFILS Jean Pierre (dét.)	Chimie
M. BIAYS Pierre	G.A.S.
M. BILLARD Jean	Physique
M. BONNOT Ernest	Biologie
M. BOUGHON Pierre	Mathématiques
M. BOURIQUET Robert	Biologie
M. CELET Paul	Sciences de la Terre
M. COEURE Gérard	Mathématiques
M. CONSTANT Eugène	I.E.E.A.
M. CORDONNIER Vincent	I.E.E.A.
M. DEBOURSE Jean Pierre	S.E.S.
M. DELATTRE Charles	Sciences de la Terre
M. DURCHON Maurice	Biologie
M. ESCAIG Bertrand	Physique
M. FAURE Robert	Mathématiques
M. FOURET René	Physique
M. GABILLARD Robert	I.E.E.A.
M. GRANELLE Jean Jacques	S.E.S.
M. GRUSON Laurent	Mathématiques
M. GUILLAUME Jean	Biologie
M. HECTOR Joseph	Mathématiques
M. HEUBEL Joseph	Chimie
M. LABLACHE COMBIER Alain	Chimie
M. LACOSTE Louis	Biologie
M. LANSRAUX Guy	Physique
M. LAVEINE Jean Pierre	Sciences de la Terre
M. LEBRUN André	C.U.E.E.P.
M. LEHMANN Daniel	Mathématiques
Mme LENOBLE Jacqueline	Physique
M. LHOMME Jean	Chimie
M. LOMBARD Jacques	S.E.S.
M. LOUCHEUX Claude	Chimie
M. LUCQUIN Michel	Chimie
M. MAILLET Pierre	S.E.S.
M. MONTREUIL Jean	Biologie
M. PAQUET Jacques	Sciences de la Terre
M. PARREAU Michel	Mathématiques
M. PROUVOST Jean	Sciences de la Terre

.../...

Professeurs 1ère classe (suite)

M. SALMER Georges	I.E.E.A.
Mme SCHWARTZ Marie Hélène	Mathématiques
M. SEGUIER Guy	I.E.E.A.
M. STANKIEWICZ François	Sciences Economiques
M TILLIEU Jacques	Physique
M. TRIDOT Gabriel	Chimie
M. VIDAL Pierre	I.E.E.A.
M. VIVIER Emile	Biologie
M. WERTHEIMER Raymond	Physique
M. ZEYTOUNIAN Radyadour	Mathématiques

P R O F E S S E U R S 2ème C L A S S E

M. AL FAKIR Sabah	Mathématiques
M. ANTOINE Philippe	Mathématiques
M. BART André	Biologie
Mme BATTIAU Yvonne	Géographie
M. BEGUIN Paul	Mathématiques
M. BELLET Jean	Physique
M. BKOUCHE Rudolphe	Mathématiques
M. BOBE Bernard	S.E.S.
M. BODART Marcel	Biologie
M. BOILLY Bénoni	Biologie
M. BONNELLE Jean Pierre	Chimie
M. BOSQ Denis	Mathématiques
M. BREZINSKI Claude	I.E.E.A.
M. BRUYELLE Pierre (Chargé d"enseignement)	Géographie
M. CAPURON Alfred	Biologie
M. CARREZ Christian	I.E.E.A.
M. CHAMLEY Hervé	E.U.D.I.L.
M. CHAPOTON Alain	C.U.E.E.P.
M. COQUERY Jean Marie	Biologie
Mme CORSIN Paule	Sciences de la Terre
M. CORTOIS Jean	Physique
M. COUTURIER Daniel	Chimie
Mle DACHARRY Monique	Géographie
M. DEBRABANT Pierre	E.U.D.I.L.
M. DEGAUQUE Pierre	I.E.E.A.
M. DELORME Pierre	Biologie
M. DEMUNTER Paul	C.U.E.E.P.
M. DE PARIS Jean-Claude	Mathématiques

.../...

M. DEVRAINNE Pierre	Chimie
M. DHAINAUT André	Biologie
M. DORMARD Serge	S.E.S.
M. DOUKHAN Jean Claude	E.U.D.I.L.
M. DUBOIS Henri	Physique
M. DUBRULLE Alain	Physique
M. DUEE Gérard	Sciences de la Terre
M. DYMENT Arthur	Mathématiques
M. FLAMME Jean Marie	E.U.D.I.L.
M. FONTAINE Hubert	Physique
M. GERVAIS Michel	S.E.S.
M. GOBLOT Rémi	Mathématiques
M. GOSSELIN Gavriel	S.E.S.
M. GOUDMAND Pierre	Chimie
M. GREVET Patrice	S.E.S.
M. GUILBAULT Pierre	Biologie
M. HANGAN Théodor	Mathématiques
M. HERMAN Maurice	Physique
M. JACOB Gérard	I.E.E.A.
M. JACOB Pierre	Mathématiques
M. JOURNEL Gérard	E.U.D.I.L.
M. KREMBEL Jean	Biologie
M. LAURENT François	I.E.E.A.
Mlle LEGRAND Denise	Mathématiques
Mlle LEGRAND Solange	Mathématiques (Calais)
Mme LEHMANN Josiane	Mathématiques
M. LEMAIRE Jean	Physique
M. LENTACKER Firmin	G.A.S.
M. LEVASSEUR Michel	I.P.A.
M. LHENAFF René	G.A.S.
M. LOCQUENEUX Robert	Physique
M. LOSFELD Joseph	I.E.E.A.
M. LOUAGE Francis	E.U.D.I.L.
M. MACKE Bruno	Physique
M. MAIZIERES Christian	I.E.E.A.
Mlle MARQUET Simone	Mathématiques
M. MESSELYN Jean	Physique
M. MIGEON Michel	E.U.D.I.L.
M. MIGNOT Fulbert	Mathématiques
M. MONTEL Marc	Physique
Mme NGUYEN VAN CHI Régine	G.A.S.
M. PARSY Fernand	Mathématiques
Mlle PAUPARDIN Colette	Biologie

Professeurs 2ème classe (suite)

M. PERROT Pierre	Chimie
M. PERTUZON Emile	Biologie
M. PONSOLLE Louis	Chimie
M. PORCHET Maurice	Biologie
M. POVY Lucien	E.U.D.I.L
M. RACZY Ladislas	I.E.E.A.
M. RICHARD Alain	Biologie
M. RIETSCH François	E.U.D.I.L.
M. ROGALSKI Marc	M.P.A.
M. ROUSSEAU Jean-Paul	Biologie
M. ROY Jean-Claude	Biologie
M. SALAMA Pierre	S.E.S.
Mme SCHWARZBACH Yvette (CCP)	M.P.A.
M. SCHAMPS Joël	Physique
M. SIMON Michel	S.E.S.
M. SLIWA Henri	Chimie
M. SOMME Jean	G.A.S.
Mle SPIK Geneviève	Biologie
M. STERBOUL François	E.U.D.I.L.
M. TAILLIEZ Roger	Institut Agricole
M. TOULOTTE Jean-Marc	I.E.E.A.
M. VANDORPE Bernard	E.U.D.I.L.
M. WALLART Francis	Chimie
M. WATERLOT Michel	Sciences de la Terre
Mme ZINN JUSTIN Nicole	M.P.A.

CHARGES DE COURS

M. TOP Géard	S.E.S.
M. ADAM Michel	S.E.S.

CHARGES DE CONFERENCES

M. DUVEAU Jacques	S.E.S.
M. HOFLACK Jacques	I.P.A.
M. LATOUCHE Serge	S.E.S.
M. MALAUSSENA DE PERNO Jean-Louis	S.E.S.
M. OPIGEZ Philippe	S.E.S.

Je tiens à remercier :

- Monsieur Christian CARREZ, Professeur à l'Université de Lille I, qui me fait l'honneur de présider le jury. Ses remarques et critiques ont été précieuses.

- Monsieur Vincent CORDONNIER, Professeur à l'Université de Lille I, qui m'a proposé ce sujet. Ses conseils et encouragements m'ont permis de faire progresser et aboutir ce travail.

- Messieurs Jean-Claude GENTINA, Professeur à l'I.D.N.
Jean-Claude POREE, Directeur Régional de l'ANVAR
de bien vouloir s'intéresser à ce travail et d'avoir accepté de l'examiner.

- Les collègues de laboratoire avec qui j'ai pu avoir de fructueuses discussions : souvent à notre insu, ils ont pu enrichir de leur influence tel ou tel aspect (ils se reconnaîtront).

- L'équipe de Reconnaissance de la Parole du CNET (Lannion), en particulier Messieurs Francis GILLET et Guy MERCIER.

Je remercie également,
pour la gentillesse et le soin qu'ils ont apportés à la réalisation matérielle du présent rapport :

- Madame Patricia CARON, qui en a effectué la dactylographie,

- Monsieur et Madame DEBOCK qui en ont assuré l'impression.

- A Véronique,

à mes parents,

à Pierre -

TABLE DES MATIÈRES

INTRODUCTION

Cette étude tente de répondre à une double préoccupation :

- les mécanismes associatifs, d'une part, en dépit de l'enthousiasme important et justifié qu'ils ont pu susciter semblent rencontrer deux difficultés essentielles :
. Le manque d'outils matériels adaptés qui leur procureraient un avantage décisif face à des solutions "classiques"), mais surtout :
. Le manque d'objectifs clairement désignés et de portée générale. Le caractère particulier des fonctions associatives devrait induire une **démarche originale**. Il faut, en effet, tenir compte assez tôt du type de problème à traiter, et reconnaître les limites effectives d'une solution associatives.

- la reconnaissance automatique de la parole (R.A.P.) d'autre part, voit ses besoins se diversifier. Parallèlement à l'investigation d'aspects fondamentaux et algorithmiques, les applications potentielles justifient le recours à des **machines spécialisées**. En particulier, l'objectif de réponse en temps réel devenant impérieux, il s'avère indispensable d'envisager des techniques matérielles évoluées. Celles-ci doivent permettre l'accélération du traitement (pipe-line, traitement parallèle, ...) et le transfert de l'"intelligence" vers le lieu de stockage (accès associatif).

Le rapprochement de ces deux domaines est alors assez naturel : en effet, parmi l'ensemble des techniques matérielles pouvant contribuer à améliorer l'implémentation des systèmes de R.A.P., le traitement associatif peut jouer un rôle privilégié. Les qualités à rechercher sont de deux types :

- puissance de calcul, permettant un traitement numérique rapide du signal de parole. Tout transfert des algorithmes vers le câblé contribue à cet aspect. En outre, aux niveaux "linguistiques" de la reconnaissance, les algorithmes les plus sophistiqués nécessitent des performances croissantes (ce point n'est toutefois pas du ressort exclusif de machines associatives),

- faculté d'"association", puisqu'une part importante de la R.A.P. (et d'autres problèmes de reconnaissance de formes) consiste à évaluer les données à identifier par rapport à des références stockées (phonèmes, mots du lexique, règles de syntaxes ...).

Contrairement à la démarche suivie habituellement, il s'agissait donc de confronter un "outil", la mémoire associative, avec une "tâche", la reconnaissance automatique de la parole, qui pouvait raisonnablement en tirer parti. A la limite, l'étude aurait débouché sur la conception d'un "processeur associatif de reconnaissance de la parole", constituant en quelque sorte, l'intersection des deux domaines.

Il fallait donc d'abord mieux définir les **machines associatives** (l'outil) et comprendre ce qui, au lieu d'en faire une panacée, leur confère un caractère original (chapitre I). De ce constat, une certaine distanciation vis-à-vis des concepts classiques s'imposait, ce qui débouche sur une démarche différente visant à décrire un fonctionnement relativement abstrait d'une mémoire associative (chapitre II). Cette **description fonctionnelle** se traduit par une **organisation logique** reposant sur une représentation simple des objets et une typologie particulière des mécanismes d'accès (chapitre III).

Outre l'influence essentielle que possède le **compromis performances/coût** par rapport à une solution classique, une solution associative matérielle peut dépendre fortement du type d'utilisation : en particulier, dans certains cas, la fonction de stockage n'est vouée qu'à la consultation, ce qui permet de considérer la mémoire associative comme une **mémoire "morte"**. Cette possibilité peut faciliter énormément l'organisation du stockage associatif.

A ce stade, apparaît alors de façon nette - contrairement au cas d'une machine à usage général - la nécessité, pour envisager précisément un système associatif, de se situer dans un contexte particulier. C'est pourquoi on essaie de discerner, dans le cadre de la R.A.P., quel(s) rôle(s) peuvent jouer les mécanismes associatifs (chapitre IV).

Il apparait alors clairement confirmé que la machine associative malgré ses possibilités, ne peut se suffire à elle-même, et qu'elle ne peut que s'insérer dans une démarche de conception plus globale.

Dans le cas de la R.A.P., l'ampleur du problème est telle que cela supposerait une tout autre approche :

- envisager le problème de la R.A.P. dans son ensemble dépasserait le cadre de cette étude.

- se concentrer sur tel ou tel aspect du problème serait plus réaliste mais imposerait au reste du système des contraintes trop rigoureuses, voire injustifiées.

A posteriori, le bilan effectué a donc permis d'élaborer une **description** différente de l'outil et une **méthodologie** pour son utilisation. Une **proposition de système** (chapitre V) sert en fait à concrétiser ce propos. Bien que délibérément plus générale, elle pourrait trouver un champ d'application privilégié dans le domaine du traitement de la parole, et, en particulier, au niveau acoustico-phonétique.

Une réalisation matérielle issue de ce travail est en cours. Elle emprunte à la présente étude une part importante de sa méthodologie et de ses conclusions.

CHAPITRE I

MACHINES ET TRAITEMENT ASSOCIATIFS

Les mécanismes associatifs consistent essentiellement en une recherche des informations d'après leur contenu. L'évaluation, pour l'ensemble des données mémorisées, du critère d'interrogation peut être envisagée sous deux angles :

- traitement parallèle, portant simultanément sur les cellules de stockage.
- traitement séquentiel, portant itérativement sur les cellules de stockage.

Les différentes propositions de machines associatives s'articulent donc entre ces deux pôles.

On peut alors distinguer trois catégories de machines associatives :

- machines parallèles du type STARAN, PEPE, ... (voir I-2.2.).
- machines séquentielles "classiques" éventuellement spécialisées en fonction d'un contexte d'application.
- mémoires associatives spécialisées dans une fonction particulière (mémoire associative de pagination, base de données associative, ...).

De ce fait, les progrès technologiques ou architecturaux mettront l'accent soit sur une extension du parallélisme, soit sur une accélération du traitement séquentiel, soit pour des considérations économiques, sur un compromis entre ces deux aspects. A ce titre, les améliorations envisageables débordent largement le cadre du traitement associatif, malgré la tendance naturelle des auteurs à réduire celles-ci à leur impact sur le domaine que nous envisageons. Cela explique certaines incursions dans des aspects plus généraux.

L'objet de ce chapitre est double :

- Présenter les concepts propres au traitement associatif, en rappelant les notions qu'on peut considérer comme "classiques". Etant donné l'étendue des publications qu'on peut rattacher à ce domaine, on a tenté d'être le plus complet possible. La difficulté est accrue par les différentes interprétations du terme "associatif" qui ont pu être employées. On pourra donc éventuellement, à la lecture, soit approfondir tel ou tel point grâce à la bibliographie, soit, au contraire, sauter telle ou telle partie.

.../...

- Exploiter le bilan de ces nombreux travaux. A priori, il est assez paradoxal de constater, à côté de cette abondance de publications, la rareté relative des applications opérationnelles. Cela justifie sans doute le caractère spécifique des contextes d'utilisation des mécanismes associatifs.

Chacun des volets de la présentation qui suit examine un angle d'approche et débouche sur une discussion critique portant sur ce point précis. Toutefois, il est évident que tous ces points sont liés. Aussi, la dernière partie (I-5) tente-t-elle de dégager une vue plus synthétique. Celle-ci tend à envisager le point de vue du concepteur de système (c'est-à-dire l'utilisateur de l'"outil" associatif) pour essayer de dégager quelle serait la démarche à envisager pour utiliser effectivement les mécanismes associatifs.

I-1 - GÉNÉRALITÉS

Il semble nécessaire dès l'abord de préciser quelques éléments de terminologie. Autrement dit, que recouvrent exactement les expressions :

. traitement associatif

. processeur associatif

. mémoire associative.

Le lexique de l'IFIP (cité par MINKER [1]) donne la définition suivante d'une **mémoire associative** : "mémoire dont les registres sont identifiés non par leur nom ou leur emplacement mais par leur contenu".

Remarquons, en ce qui concerne le **traitement associatif** que tout langage de programmation relativement évolué fournit à l'utilisateur des mécanismes de type associatif. Ainsi, l'interrogation d'une banque de données revient à formuler des questions du style : "enregistrement du CLIENT domicilié à LILLE dont une facture de **plus de 3 mois** est **impayée**". On recourt bien dans ce cas à une partie du **contenu** de l'enregistrement pour retrouver l'enregistrement complet. D'autres exemples montreraient que ce genre de mécanismes est très répandu. Nous admettrons qu'on peut distinguer des mécanismes logiciels, que nous examinerons séparément, et des mécanismes matériels.

Notre première restriction est donc d'admettre qu'une mémoire associative est un ensemble **câblé** de registres auxquels l'accès s'effectue en énonçant tout ou partie de leur contenu.

On pourrait, comme PARHAMI [2], s'en tenir là et se contenter de considérer la mémoire associative comme une "boîte noire" dont on ne connaît que le mode de fonctionnement décrit plus haut. Les mécanismes internes qui permettent de réaliser cette fonction sont alors indifférents.

Cependant de nombreux auteurs admettent (par exemple : YAU & FUNG [3]) que l'accès par le contenu implique, pour des raisons d'efficacité, un examen en **parallèle** de tous les registres, en une seule opération. Mieux on peut aller jusqu'à envisager des traitements plus complexes, sur un plan global.

Une mémoire associative se caractérise donc alors par les deux aspects suivants (CORDONNIER [4]) :

- **accès à l'information par le contenu** (ou une partie de ce contenu)

- **opérations en parallèle sur l'ensemble des données.**

On a donc, le plus souvent, relié les mémoires associatives au traitement parallèle. En effet, puisqu'une seule opération s'effectue sur la totalité des données, on a affaire à une machine de type SIMD (Single Instruction-stream, Multiple Data-stream) selon la classification introduite par FLYNN [5].

Cependant, si certaines machines, comme par exemple STARAN (G. SAYRE [6]), recouvrent bien les deux domaines, ceux-ci ne sont pas confondus : de nombreuses machines de type SIMD recourent à l'adressage pour accéder aux données. Il en est ainsi, par exemple, dans le plus connu des "processeurs-tableaux", ILLIAC IV (BARNES et al. [7]).

Les deux modes d'organisation évoqués (parallélisme ou non) ne sont pas incompatibles. Néanmoins, la définition de PARHAMI [2] semble plus générale : il ne préconise pas que l'examen des données se fasse en parallèle plutôt que séquentiellement. On notera que, selon l'échelle d'observation adoptée, un algorithme séquentiel de recherche peut être considéré comme étant parallèle si on le regarde plus globalement.

En ce qui concerne la différence entre mémoire associative et processeur associatif, nous conviendrons, comme YAU et FUNG [3], que le **processeur associatif** effectue sur une mémoire associative des traitements plus élaborés que les opérations usuelles d'échange entre un processeur et une mémoire. Toutefois, dans bien des cas, la frontière entre les deux est difficile à tracer, ainsi que le remarque PARHAMI [2]. Bien souvent les deux termes sont employés l'un pour l'autre . On pourrait donner la préférence à l'un ou l'autre selon que l'on considère plus particulièrement la fonction de mémorisation ou le traitement. Dans la suite, nous ne chercherons pas à établir de distinction subtile, et les deux termes seront utilisés pratiquement indifféremment.

Notons tout de même que l'on réserve généralement le terme "processeur" à un organe "maître" d'une exécution (processus), la mémoire étant une ressource "esclave". Ces deux points de vue peuvent éventuellement servir de critère distinctif.

En résumé, nous dirons que le **traitement associatif** consiste à accéder aux informations d'après leur contenu : dès lors, rien n'interdit d'effectuer cette fonction sur une machine classique en se donnant les outils logiciels appropriés.

Un **processeur associatif** sera une machine capable de supporter directement les mécanismes associatifs ; elle contiendra nécessairement une **mémoire associative**, telle qu'on l'a définie plus haut.

Pour en terminer avec le vocabulaire, notons qu'on a également désigné les mémoires associatives par les noms de : mémoire adressable par le contenu, mémoire catalogue, mémoire à interrogation parallèle, mémoire à logique distribuée, etc ...

Le schéma qui suit (figure 1) tente de représenter les divers niveaux auxquels on peut intervenir ; comme on l'a dit, il est tout à fait possible d'effectuer des fonctions associatives sur une machine classique. Réciproquement, rien n'interdit d'utiliser un processeur associatif pour des traitements "classiques". Cependant, et nous reviendrons sur ce point plus loin, l'utilisation en tant que machine universelle d'un processeur associatif est sujette à critique.

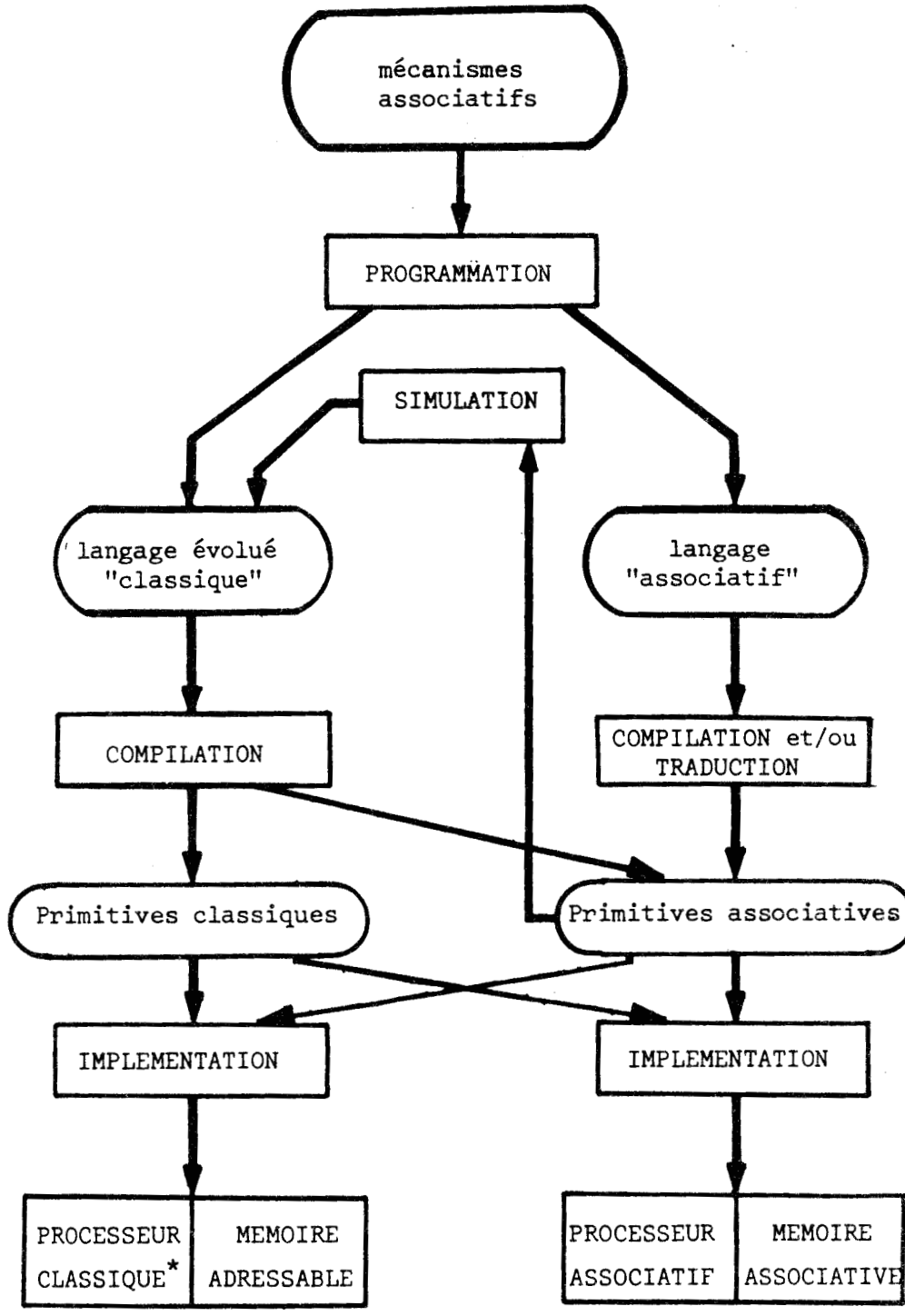


Fig 1 : Multiplicité des approches du TRAITEMENT ASSOCIATIF

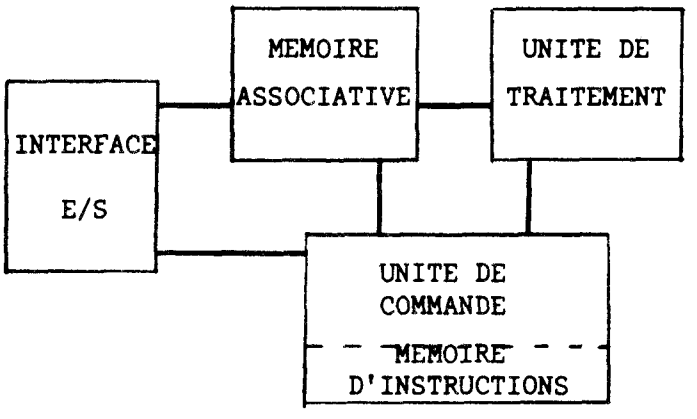
* N.B. : On définit ici principalement un processeur "classique" par référence à la nécessité d'adressage. C'est un point de vue subjectif mais il permet d'illustrer les différents niveaux d'intervention du concept associatif.

I-2 - ASPECTS ARCHITECTURAUX

On décrira tout d'abord quels sont les **éléments fonctionnels** qui composent généralement un processeur associatif. Ensuite, comme l'organisation de la mémoire associative elle-même a une incidence primordiale sur l'architecture du processeur, les divers **modes d'organisation** seront examinés.

I-2.1. Architecture d'un processeur associatif

La plupart des processeurs associatifs peuvent se décrire de la façon suivante (YAU & FUNG [3]) :



La différence essentielle par rapport à une machine classique est l'utilisation d'une **mémoire associative**. Elle implique d'autres changements dans d'autres parties du processeur (absence du mécanisme d'adressage notamment). En outre, les **traitements** arithmétiques et logiques (ou une partie d'entre eux) peuvent s'effectuer **en parallèle** sur l'ensemble des données ou un sous-ensemble déterminé de celles-ci.

Il convient ici de remarquer une caractéristique importante, rarement évoquée de façon explicite dans la littérature : dans l'hypothèse de l'utilisation d'un processeur associatif en tant que machine universelle (utilisation équivalente d'une machine de Von Neumann), une séparation entre mémoire de programme et mémoire de données semble souhaitable. En effet, il ne paraît pas avantageux, comme le remarque MICHEL [8], de

stocker un programme autrement que dans une mémoire séquentielle ; cela, du moins, tant que l'algorithmique et les langages imposent aux programmes une structure relativement séquentielle. En fait, ce problème nécessite peut être de prendre plus de recul vis-à-vis des machines de type Von Neumann. L'alternative serait alors la suivante :

- doit-on (peut-on) utiliser un processeur associatif à la place d'une machine adressable.

- doit-on développer des applications, des langages, une algorithmique adaptés aux processeurs associatifs.

Nous reviendrons sur ce point en examinant l'utilisation des machines associatives.

Quelle que soit l'organisation adoptée, les mêmes fonctions sont nécessaires et la mémoire associative comprendra donc les éléments suivants (figure 2) :

. le réseau-mémoire : assure la **fonction de stockage** des informations proprement dite. "Réseau" est ici à prendre dans un sens relativement large.

. le registre comparande* : contient l'information qui sert de critère d'accès ; il peut éventuellement servir de registre de transfert entre la mémoire et le processeur, selon l'organisation adoptée. Il assure la **fonction d'interrogation.**

. le registre masque : permet de sélectionner certains champs du registre précédent et donc de préciser ou modifier le critère d'accès. Ce mécanisme complète donc le précédent dans la réalisation de la **fonction d'interrogation.**

. la logique de comparaison : elle exécute les opérations de recherche suivant les ordres reçus et **décodés** par le processeur associatif. Le degré de **parallèlisme** de cette opération caractérise un processeur associatif.

. le réseau d'indicateurs : il s'agit de bascules associées aux positions de stockage et permettant de marquer celles qui satisfont à une interrogation.

* *comparande : un des opérandes sur lesquels porte une comparaison.*

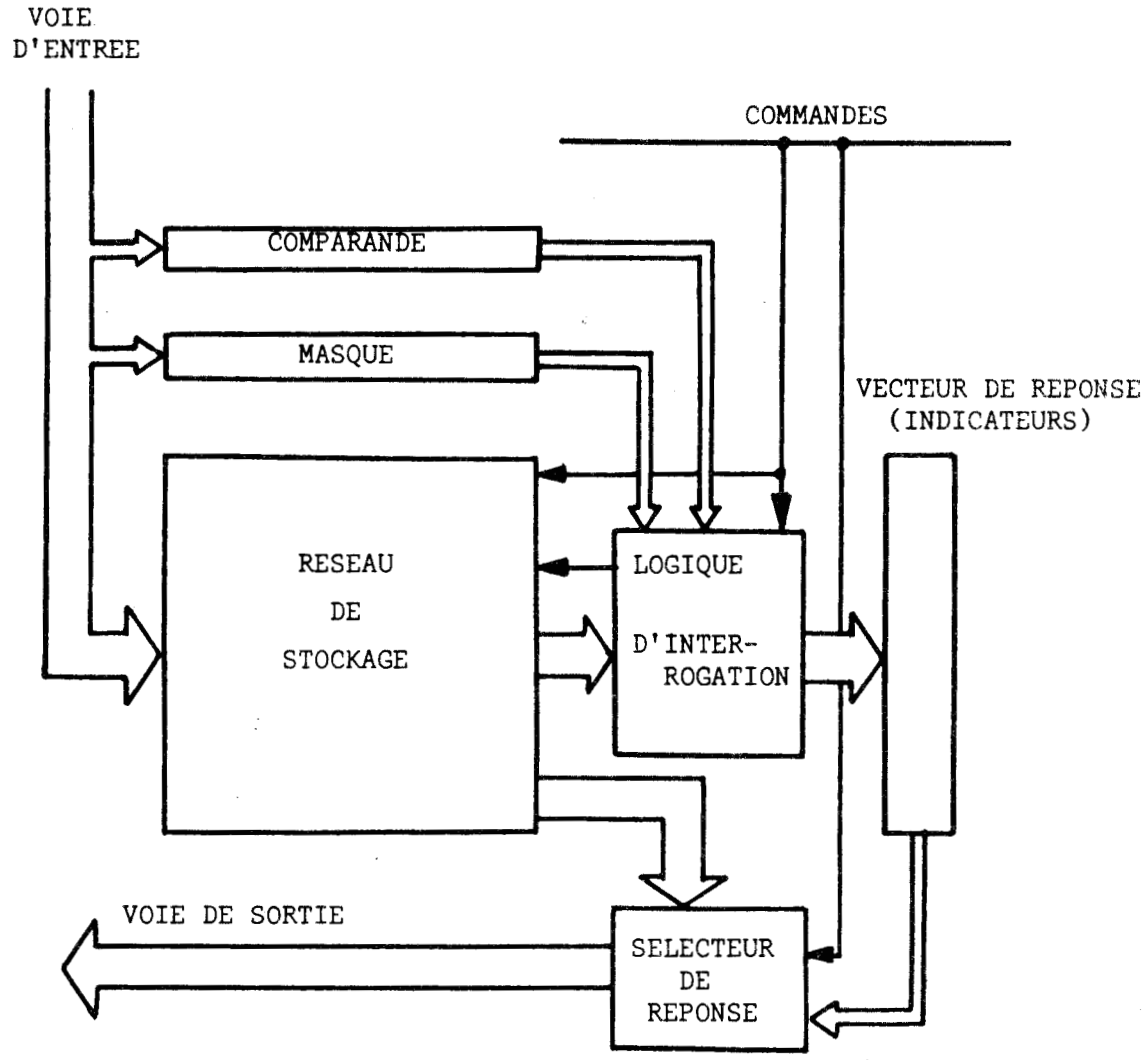


Fig. 2 : Eléments d'une mémoire associative

Ils peuvent également servir à indiquer les emplacements disponibles pour l'écriture, des cas d'erreurs, etc ... Pour chaque évènement considéré, ces indicateurs effectuent la **fonction de repérage**. Ils servent de lien entre des opérations successives.

. le sélecteur de réponses : il s'agit de résoudre le problème des réponses multiples pour pouvoir choisir une ou plusieurs informations satisfaisant au critère d'interrogation. Pour cela, il faut déterminer une stratégie qui permette de définir les organes câblés ou les algorithmes nécessaires. Nous reviendrons plus loin sur ce problème qui recouvre en fait la **fonction de sélection** de l'information.

Ces éléments, et les fonctions qu'ils assurent peuvent éventuellement subir des variantes, selon les réalisations.

La répartition réelle des possibilités de traitement parmi les cellules de la mémoire de façon à assurer le parallélisme implique des solutions nombreuses et diverses.

Cependant, on peut admettre, qu'au moins fonctionnellement, toutes les architectures seront du type S.I.M.D. : il suffira d'observer le processeur associatif à un niveau convenable dans l'espace ou pendant une durée suffisante dans le temps.

I-2.2. Organisations associatives

Les diverses organisations possibles des processeurs associatifs (en fait, comme on l'a vu, des mémoires associatives) se regroupent en quatre catégories (PARHAMI [2]). Cette classification est adoptée par la plupart des auteurs (notamment YAU et FUNG [3]).

Il est hors de propos de décrire en détail ces organisations ; pour cela on se reportera au travail de YAU et FUNG [3]. Nous nous contenterons de décrire les principes de base de chaque type d'organisation et de citer, si possible un ou des exemples significatifs. En effet, il semble plus important de connaître les fonctions possibles, les utilisations que de s'attarder sur des caractéristiques dont l'impact concerne principalement les deux termes du compromis coût/performances. En fait, les

différences essentielles entre ces organisations toucheront la rapidité de réponse et la capacité.
On distingue donc, selon le mode de comparaison et d'accès aux cellules de la mémoire, des systèmes :

- totalement parallèles . par mots
  . à logique distribuée
- séquentiels par bits
- séquentiels par mots
- structurés par blocs.

I-2.2.1. *Systèmes totalement parallèles*

. Systèmes totalement parallèles par mots (figure 3)

Chaque cellule de la mémoire dispose d'une unité logique propre (processeur élémentaire). Les opérations possibles sont très limitées. Mais le coût en matériel d'une telle organisation la rend **prohibitive. Elle est** théoriquement la plus rapide. Les espoirs fondés sur les techniques cryogéniques (voir § I-3) laissaient entrevoir des possibilités mais ont été mis de côté en raison de l'importance des coûts nécessaires. On peut également citer la proposition de logique cellulaire de YANG et YAU [9].

. Systèmes totalement parallèles à logique distribuée (figure 4)

Chaque cellule comprend un caractère, la logique de comparaison associée et un indicateur d'état. Les cellules communiquent entre voisines et il est possible de travailler sur un groupe de cellules, ce qui permet de définir des articles de taille variable.

Ce genre d'organisation semble bien adapté à la gestion de fichiers.

La première machine connue est celle proposée par LEE & PAULL [10]. Elle était principalement destinée à la recherche d'informations alphanumériques. Des modifications ou améliorations successives ont été proposées.

Plus récemment, on peut noter principalement la "mémoire intelligente" d'EDELBERG & SCHISSLER [11], et PEPE, développé aux Bell Laboratories (cf. notamment H.G. MARTIN [12]).

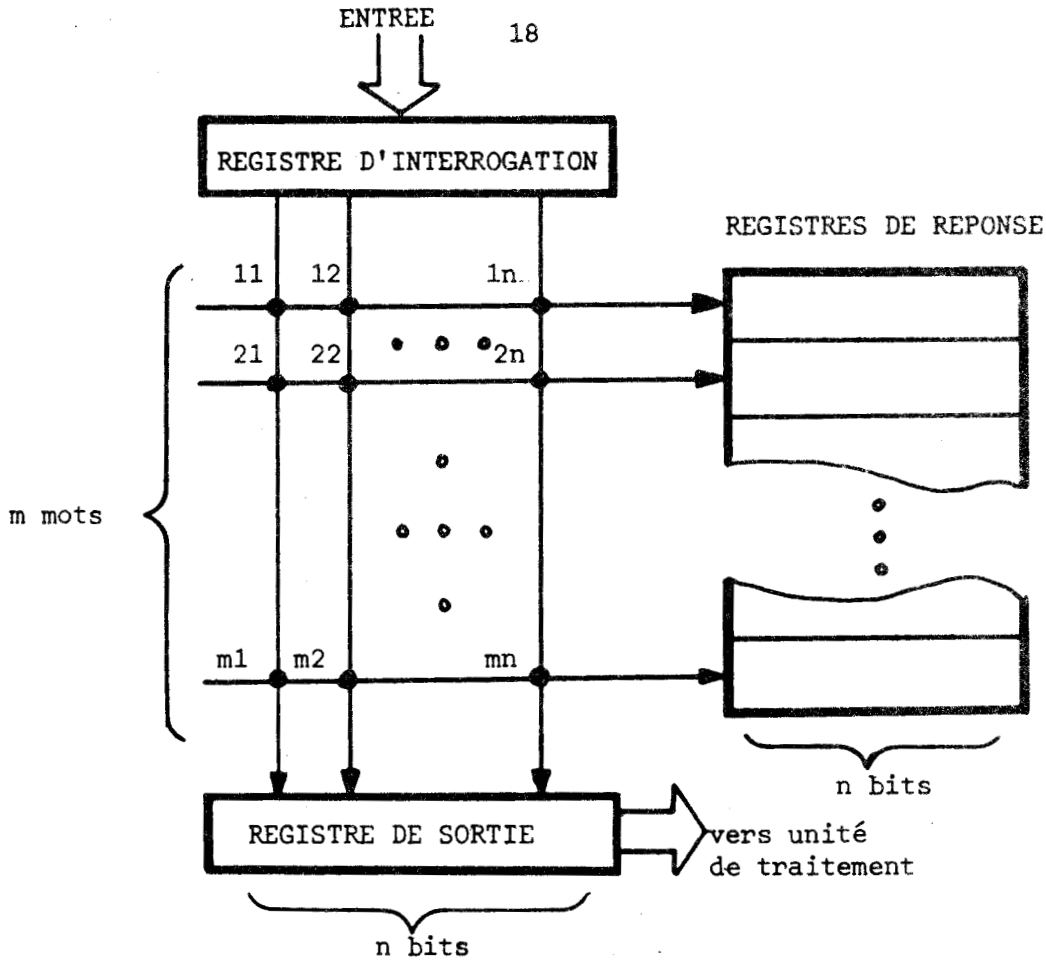


Fig 3 : Mémoire associative totalement parallèle

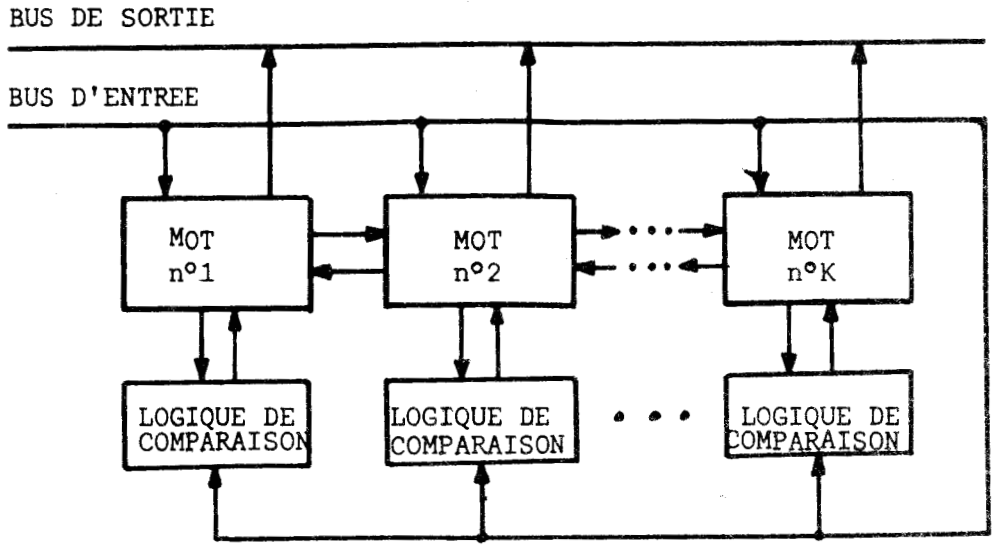


Fig. 4 : Mémoire associative parallèle à logique distribuée

La "mémoire intelligente" se compose de boucles synchrones séparées par des processeurs élémentaires (logique distribuée). Ceux-ci peuvent soit travailler sur une boucle fermée, soit échanger le contenu de 2 boucles. La configuration (longueur des boucles) est modifiable dynamiquement.

PEPE (THURBER [13]) est l'un des deux gros calculateurs associatifs effectivement implémentés. Il se compose d'un certain nombre (variable) de processeurs élémentaires reliés à travers un système de communication à un système de contrôle. Ce dernier est connecté à un CDC 7600 qui lui sert de calculateur-hôte. Les processeurs élémentaires comprennent :

- une mémoire locale de 1024 mots de 32 bits
- une unité arithmétique
- une unité de corrélation
- une unité de sortie associative.

La charge globale du système est répartie sur les P.E.s qui ont chacun la responsabilité d'un fichier qu'ils tiennent à jour en temps réel.

I-2.2.2. *Systèmes séquentiels par bits* (figure 5)

Le coût élevé en possibilités de traitement des solutions totalement parallèles a amené à ne traiter qu'un seul bit à la fois dans tous les mots simultanément. Ce genre de systèmes est donc souvent désigné comme "série par bits, parallèles par mots" ou encore "bit-slice".

Les machines réalisées sont plus nombreuses que dans le cas précédent : citons notamment OMEN, ALAP (FINNILA [14]), ECAM (ANDERSON [15]), et surtout STARAN (SAYRE [6]), dont plusieurs exemplaires, construits par Goodyear Aerospace Corp., sont en fonctionnement dans des administrations américaines (U.S.A.F., Defense Mapping Agency, NASA, ...)

ALAP (Associative Linear Array Processor) est un réseau linéaire de cellules contenant un mot et une U.A.L. élémentaire opérant en série sur ce mot. Outre des bus de données et de commande communs, les cellules communiquent entre elles par un canal de chaînage. Cela permet aux cellules de résoudre des conflits dus aux réponses multiples, de faire certaines opérations englobant plusieurs mots, d'effectuer des tris, ... En fait, ALAP semble axé sur la partie traitement plutôt que sur la partie stockage.

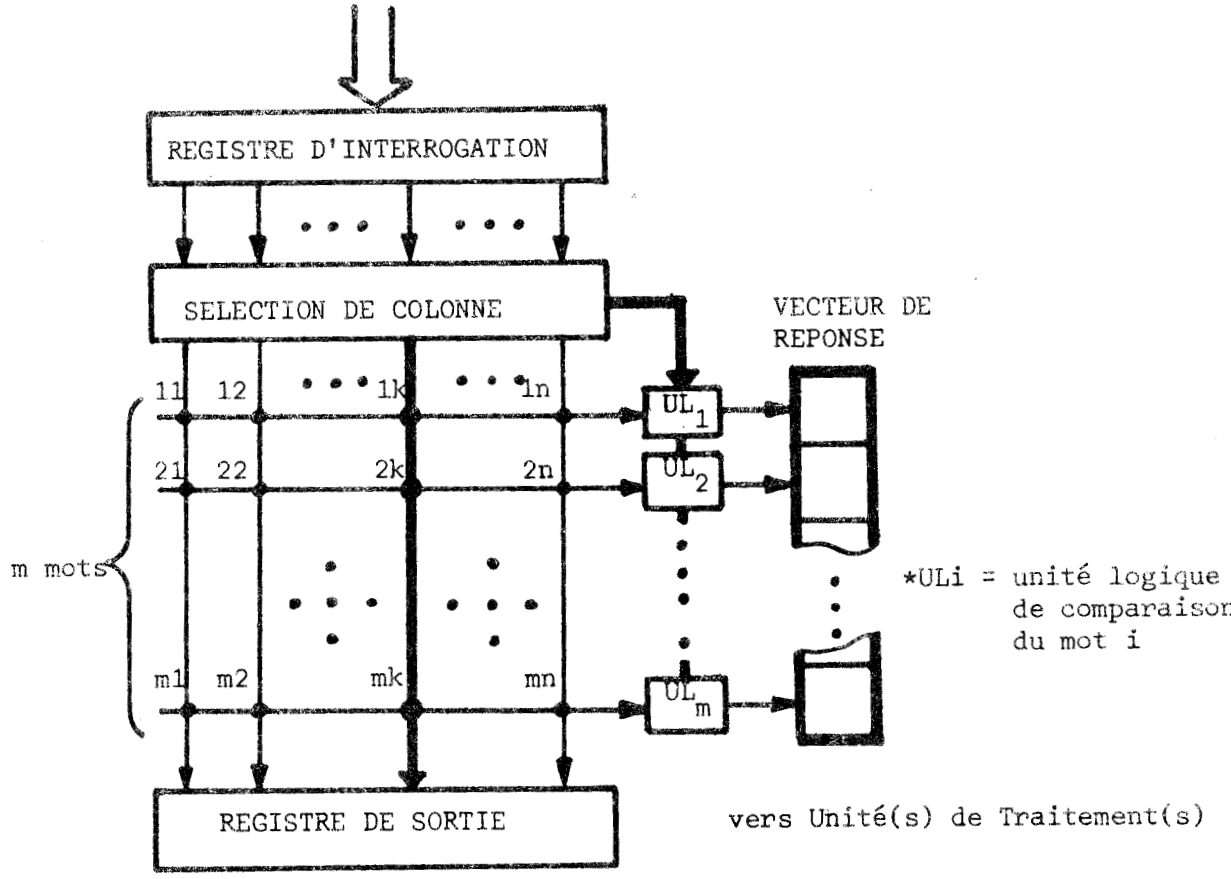


Fig. 5 : Mémoire associative séquentielle/bits,parallèle/mots

ECAM (Extended Content Addressed Memory) se compose d'un réseau de mémoire associative (CAM) et d'une unité de contrôle. Le réseau CAM est constitué de cellules de 4096 bits organisés en 16 registres à décalage de 256 bits avec accès aléatoire à l'un de ces registres.

Dans STARAN (THURBER [16]) les modules de mémoire associative sont des réseaux de 256 mots de 256 bits, avec 256 processeurs élémentaires, un réseau de permutation et un sélecteur. Les processeurs travaillent sur un mot bit par bit. Le réseau de permutation est l'élément essentiel puisqu'il permet d'accéder au réseau mémoire par tranche de bits, par mots ou une combinaison de ces 2 accès. En outre, il permet, pour faciliter les opérations logiques, arithmétiques, ou de recherche entre les mots, de décaler ou réorganiser les mots du réseau.

I-2.2.3. *Remarque : Interconnexion entre processeurs et cellules de mémoire*

La plupart de ces machines sont plus axées vers le parallélisme que sur l'utilisation des mécanismes associatifs. Mis à part le cas d'ALAP qui constitue pratiquement une machine à logique distribuée, ce sont surtout des **processeurs de type "tableaux"** : à chaque instruction globale (car ces machines prennent en charge des traitements de type S.I.M.D.) se pose le problème de la connexion d'un processeur élémentaire au sous-ensemble de données qui lui est affecté (exemples : contrôle aérien, météorologie, calcul matriciel, ...). Cela déborde le cadre du traitement associatif pour se poser généralement à toute machine de ce type (par exemple dans ILLIAC IV).

Dans STARAN (BATCHER [17]) le réseau de permutation jouait ce rôle.

Plus généralement, il s'agit pratiquement d'**anticiper,** en préparant les données appropriées par des **réorganisations** telles que tris, permutations etc ... D'où l'idée de confier ce travail à un organe spécialisé et performant. Ainsi FENG [18] a proposé la notion de "manipulateur de données" : il s'agit donc d'un processeur d'interconnexion entre les cellules de stockage et les processeurs d'un système de type S.I.M.D. Les possibilités de manipulations peuvent être nombreuses : elles peuvent servir à des permutations (décalages, échanges, mélanges, tris, fusions, ...) à des recopies (doubles, multiples ..)

à des gestions d'espace (compression, expansion, transfert), à des masquages, des complémentations, etc ...

Il s'agit d'une des nombreuses possibilités d'interconnexion dans un système S.I.M.D.

Dès qu'un certain degré de séquentialité apparaît dans le système (pour des raisons de coût et/ou de capacité en général) il est intéressant de prévoir des mécanismes d'accélération ou de **réarrangement des données** :ceux-ci permettent de réduire l'impact des cas les plus défavorables sur le temps de réponse.

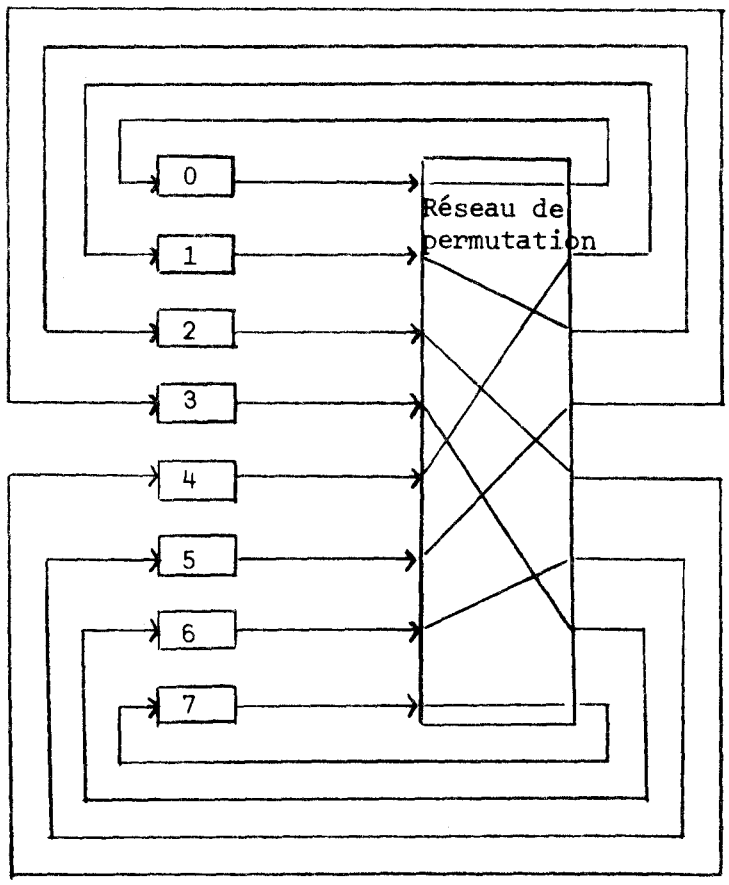
Des techniques telles que le "perfect shuffle" de **STONE** [19] ont été proposées. Ces techniques reposent principalement sur les résultats des théories mathématiques sur les **permutations**. Rappelons que cette permutation sur un vecteur de N éléments, consiste à interclasser parfaitement la première moitié du vecteur avec la seconde. Cela se définit donc de la façon suivante si l'on indexe les éléments de 0 à N-1 :

$$S(i) = 2i \quad \underline{si}\ 0 \leq i \leq \frac{N}{2} - 1$$

$$= 2i+1-N \quad \underline{si}\ \frac{N}{2} \leq i \leq N-1$$

où S(i) est le numéro de la cellule qui reçoit le contenu de la cellule i.

Ainsi, pour N = 8, on a le réseau suivant :



L'autre permutation utilisée est plus classique : il s'agit de l'échange entre 2 cellules voisines.
LAWRIE [20] a présenté un réseau d'interconnexion basé sur le "perfect shuffle" et permettant de réaliser des permutations de $N = 2^n$ éléments en $\log_2 N$ étapes. Cependant des conflits peuvent dans certains cas ôter la possibilité de permutation. En compliquant encore un peu le réseau, LANG [21] a montré la possibilité de réaliser n'importe quelle permutation en un nombre d'étapes au pire égal à $O(\sqrt{N})$. En outre, la procédure proposée améliore également les cas où n'apparaissaient pas de conflits. Précisons toutefois que **toute amélioration de ces algorithmes de permutation ne peut se faire qu'au prix d'un accroissement de complexité du matériel.**

I-2.2.4. *Systèmes séquentiels par mots* (figure 6)

On les appelle aussi "mots-série, bits parallèles". Ici encore, il s'agit de s'affranchir du coût élevé d'un système totalement parallèle, principalement dans le cas où la **capacité** nécessaire est relativement importante. La solution adoptée est en quelque sorte orthogonale à la précédente.

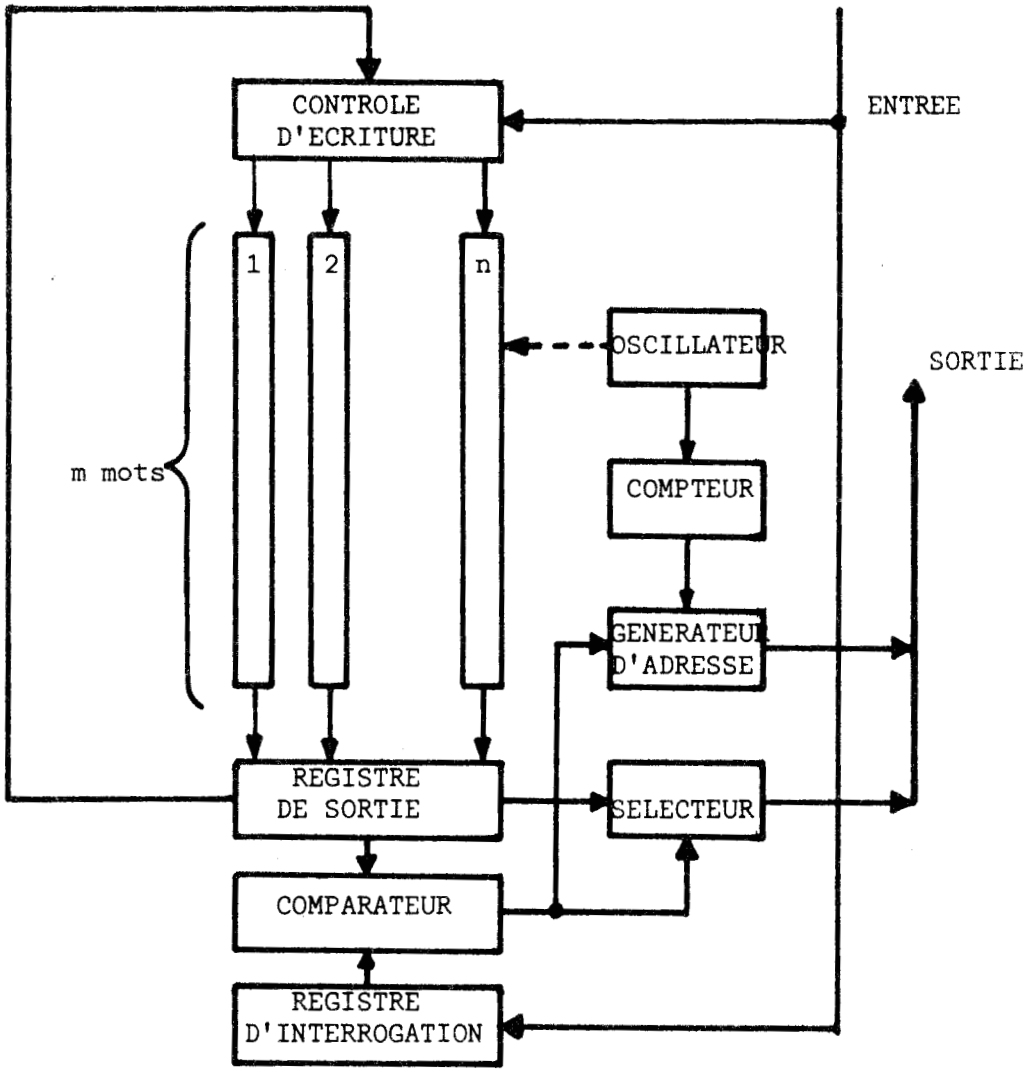


Fig. 6 : Mémoire associative séquentielle/mots (parallèle/bits)

N.B. Les dispositifs physiques dans lesquels circulent chaque bit des m mots peuvent être des lignes à retard, des registres à décalage électroniques, etc ...

D'un point de vue fonctionnel, le traitement en parallèle de tous les mots, de façon câblée, n'est pas indispensable : on peut admettre de traiter les mots de façon séquentielle : le processeur doit donc être capable de traiter la largeur d'un mot à la fois, mais il est unique (pour une séquence de mots).

Il s'agit en fait de l'implémentation câblée d'une boucle programmée de recherche séquentielle. Pour être intéressante, cette solution doit présenter des avantages par rapport à une solution programmée ; ces avantages sont essentiellement :

- temps de décodage d'instructions réduit
- utilisation possible d'un support à débit élevé (mémoire circulante par exemple) puisque la seule opération d'adressage est : "mot suivant".

Cependant, il convient de remarquer que le temps de circulation est une contrainte primordiale et que cela est d'autant plus critique que le nombre de mots est élevé.

C'est pourquoi, jusque récemment, un petit nombre de travaux ont suivi cette voie. On peut citer :

. l'utilisation de lignes à retard à ultrasons proposée par CROFUT & SOTTILE [22] (voir I-3.2)

. la mémoire connectée au calculateur NEBULA (cf. RUX [23]) : elle se composait de 35 lignes à retard en verre de 2048 bits se propageant à 20,48 MHz. Cela donnait donc une mémoire de 2048 mots de 32 bits (les 3 restants servaient à des fonctions spéciales) qui pouvaient être examinés en 100 µs. Notons que, ce système étant en quelque sorte le dual du précédent, des méthodes d'accélération de l'accès peuvent être envisagées. Les techniques de permutation, notamment, offrent, comme dans le cas des organisations séquentielles par bits, des possibilités d'amélioration du temps de réponse. Cependant elles impliquent alors la connaissance, par le processeur d'examen, des positions des informations dans les cellules. Même s'il ne s'agit pas de position absolue, leur position relative est au moins nécessaire. La gestion de la position de chaque cellule dans la boucle séquentielle, bien qu'analogue à un simple comptage, ne peut qu'alourdir les mécanismes de fonctionnement de la mémoire associative. En outre des difficultés apparaissent inévitablement au niveau des mises à jour.

Ce genre d'organisation est donc fortement limité en rapidité et en capacité en raison des difficultés technologiques. L'essor récent de mémoires séquentielles plus performantes et reposant sur des techniques de fabrication analogues aux semiconducteurs laisse présager un regain d'intérêt pour ce type de solution : en effet il semble que des technologies telles que les mémoires CCD et surtout à bulles magnétiques soient appelées à combler la lacune qui avait fait tomber en désuétude ce genre de solution.

I-2.2.5. *Systèmes organisés en blocs* (figure 7)

Dans le cas où c'est une **capacité importante** qui est la principale contrainte, les solutions précédentes présentent des inconvénients :

- les systèmes totalement parallèles ont un coût prohibitif et leur complexité matérielle les rend peu fiables.
- les systèmes séquentiels par mots sont certes plus économiques, mais au prix d'une lenteur d'autant plus gênante que le nombre de mots est élevé.
- les systèmes séquentiels par bits sont limités en nombre de mots pour des raisons de coût (un processeur élémentaire par mot) ; en taille des mots pour des raisons de rapidité.

L'organisation en blocs tente d'effectuer un compromis entre ces inconvénients.

Pour respecter l'impératif de capacité ce sont des **mémoires de masse** qui ont d'abord été choisies : il s'agit en fait de doter une mémoire de masse (disque à têtes fixes ou tambour, par exemple) de possibilités associatives.

SLOTNICK [24] a introduit le concept de système à logique-par-piste. Comme d'autres supports (électroniques notamment) sont envisageables, il est préférable de parler de systèmes à logique cellulaire (SU [25]).

La conjugaison de cette idée avec le modèle relationnel de CODD [26] a été à la base du développement des **machines bases de données**. C'est-à-dire que ces "processeurs bases de données" se trouvent au confluent des travaux des concepteurs de bases de données et de ceux exploitant le concept associatif. Toutefois certains de ces processeurs acceptent d'autres modèles de bases de données (hiérarchisées, réseaux).

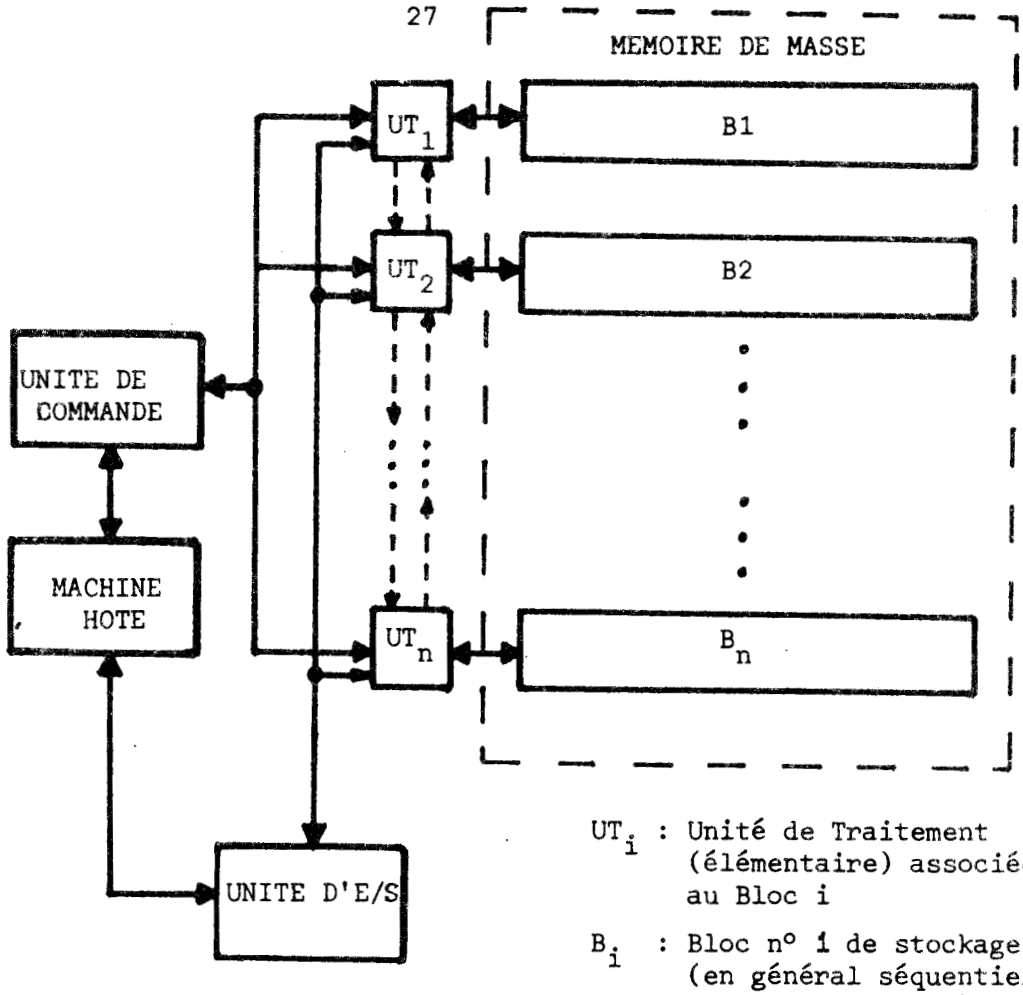


Fig. 7 : Mémoire associative organisée en blocs

Ce genre de systèmes, associant une logique distribuée (on rejoint ici les concepts de LEE) à une mémoire rotative de capacité importante, laisse entrevoir des solutions plus réalistes.Toutefois, il s'agira essentiellement de machines "backend", c'est-à-dire intégrées dans un système et n'assurant que certaines fonctions spécialisées.

Il serait fastidieux de donner une liste exhaustive des processeurs bases de données proposés : on se reportera à des articles faisant le point sur la question (BANDURSKI [27], SU [25], SMITH [28]). Nous évoquerons ici les travaux qui semblent les plus intéressants.

P.B. BERRA et E. OLIVER [29] ont étudié des configurations possibles de mémoires de masse connectées à des machines du type STARAN. On a toutefois déjà parlé du coût élevé des machines "tranches de bits" et de leur relative lenteur.

Le système CASSM (Context Addressed Segment Sequential Memory) est capable de supporter des bases de données relationnelles ou hiérarchisées (SU [25], LIPOVSKI [30] [31]). Le principe est de doter un support de mémoire rotative de têtes de lecture et d'écriture distinctes. Le délai rotationnel entre les 2 têtes est mis à profit par des processeurs élémentaires pour traiter les données. La structure des données est assez complexe (voir LIPOVSKI [31]). Le premier prototype implémenté utilisait des disques souples à têtes fixes.

Un autre système,RAP (Relational Associative Processor), était voué aux bases de données relationnelles (SADOWSKI & SCHUSTER [32]). Une particularité intéressante est que le prototype utilisait des registres à décalage CCD.

Le système RARES, à la différence des 2 précédents, n'a pas vu le jour en tant que prototype. RARES (Rotating Associative RElational Store) était destiné à des tâches spécialisées telles que le tri (LIN [33]). D'autre part,contrairement à RAP et CASSM où les enregistrements étaient stockés le long des pistes, dans RARES les relations étaient disposées transversalement (SMITH [28]).

La "mémoire intelligente" d'EDELBERG et SCHISSLER [11], dont nous avons déjà parlé, pourrait également se rattacher à ce genre d'applications.

Le DBC (Data Base Computer) proposé par HSIAO [34] est destiné à des bases de données très volumineuses. Il utilise alors des disques à têtes mobiles et permet de supporter différents modèles de bases de données (relationnelles, hiérarchisées, réseaux, ...). Un aspect original est qu'il assume les problèmes de protection des informations : un des 7 processeurs spécialisés du système a pour rôle de contrôler les accès.

L'AFP (Associative File Processor) construit par BIRD et al. [35], utilise un matériel de base conventionnel : le processeur de contrôle est un PDP-11/45 de DEC et le stockage repose sur des unités IBM 3350 (haute densité, têtes mobiles). Il permet d'effectuer de 30 à 50 demandes simultanées sur une base de données de l'ordre d'un milliard de caractères, avec un temps de réponse de 4 à 5 minutes.

Plus récemment, le constructeur anglais ICL a annoncé [36] [37] pour le courant de 1980 le système CAFS (Content Addressable File Store). Un mini-ordinateur (mots de 16 bits, capacité en mémoire centrale : 64 K octets) est associé à des unités de disques dans lesquelles on a rajouté des têtes de façon à pouvoir lire jusqu'à 10 pistes simultanément. La capacité pourrait atteindre 840 millions de caractères. Le système serait d'autant plus rentable, par rapport à un système conventionnel, que le nombre d'accès simultanés est grand (plus de 20 utilisateurs) et que les critères sont complexes. Dans ces conditions, on atteindrait des performances de 50 à 100 fois meilleures que sur un système classique. Ainsi, par exemple, les postes britanniques ont mis sur pied une expérimentation de ce système pour les renseignements téléphoniques ; pour 6 millions d'abonnés le temps de réponse est inférieur à 2 secondes .

Par ailleurs, des travaux se sont situés à un niveau intermédiaire dans la hiérarchie des mémoires. Il s'agit en fait de mettre en oeuvre les nouvelles technologies de mémoires séquentielles électroniques, notamment bulles magnétiques et CCD.

Deux voies sont alors envisageables :

La première consiste à transposer purement et simplement les modèles précédents à ces technologies. C'est l'approche de CHANG [38] par exemple, qui propose d'implémenter des bases de données relationnelles à l'aide de mémoires à bulles magnétiques. Dans le même ordre d'idées, RAP, que l'on a évoqué plus haut, utilisait des registres CCD (transferts de charges capacitives).

Nous reviendrons plus en détail sur les perspectives liées à ces techniques en abordant les aspects matériels et technologiques des mémoires associatives.

Le développement en direction des processeurs bases de données associatives (relationnelles ou non) de ces systèmes organisés en blocs leur confère une vocation fonctionnelle originale : en effet, les autres organisations tendaient à se substituer aux mémoires principales classiques : capacité limitée, temps d'accès optimal. Ce dernier type d'organisation, contrairement aux autres, évolue donc de façon tout à fait indépendante et c'est pourquoi il constitue une sorte de domaine spécifique. Cela explique la quantité remarquable de travaux qu'il suscite et qui semble presque démesurée si on la compare aux autres types d'organisations associatives.

Les systèmes informatiques manipulent des quantités de données de plus en plus considérables. Les organes de communication (canaux par exemple) entre le processeur central et les supports de stockage de masse deviennent alors de véritables goulots d'étranglement. La possibilité de doter le système de stockage d'une faculté de pré-traitement et de sélection de l'information devient indispensable : elle permet de s'affranchir de transferts fastidieux et inutiles des données et donc d'améliorer dans une large mesure l'efficacité de la recherche de l'information. De plus, le transfert d'"intelligence" depuis le processeur central vers les supports de stockage permet de consacrer celui-ci à des tâches plus nobles que la recherche d'informations.

I-2.2.6. *Critères de choix d'une architecture*

En fait, dans ce qui vient d'être décrit, on peut dégager deux aspects :

- d'une part, un problème d'**organisation de la mémoire**. Les solutions envisageables sont alors des compromis entre les contraintes technologiques et les performances souhaitées. On a vu que les degrés respectifs de parallélisme et de séquentialité, ainsi que la répartition plus ou moins décentralisée des organes de traitement constituent le facteur essentiel qui tendra à privilégier l'une ou l'autre de ces solutions.

- d'autre part, **le point de vue fonctionnel**. Il s'agit plus ici de se préoccuper de la vocation du système. On a déjà insisté sur la spécificité des machines bases de données. En fait, cela pourrait se résumer à poser la question suivante : le système envisagé se substitue-t-il parfaitement à un processeur classique, ou à l'un des niveaux d'une hiérarchie de mémoire (et dans ce cas, lequel ?).

Compte tenu de ces deux aspects, on peut déduire, de façon simplifiée, que le choix d'une des architectures décrites repose essentiellement sur deux contraintes :

- la **capacité** des informations qui constituent l'environnement des problèmes que le système sera amené à traiter.
- le **temps de réponse** espéré.

Il faut alors distinguer la capacité virtuelle globale de la capacité effectivement utile à un instant donné (c'est-à-dire le contexte). De même, le temps de réponse sera évidemment proportionné en fonction du type de demande.

Remarquons que ces deux contraintes sont directement imposées par la nature de l'application envisagée : ainsi la notion de "temps réel", par exemple, n'est pas appréciée de façon identique si l'on traite l'accès interactif à une base de données dans une application de type guichet administratif, ou si l'on traite la surveillance radar d'une zone aérienne. Cela semble corroborer l'opinion intuitive qui a été énoncée plus haut : les processeurs associatifs ne semblent pas devoir se substituer aux machines universelles, mais au contraire paraissent devoir s'orienter vers des applications spécifiques.

A ce stade de la discussion, nous avons - sans l'admettre explicitement - supposé acquis le choix d'une implémentation câblée de mécanismes associatifs. Or, la nécessité d'un tel choix doit être préalablement évaluée : il est parfaitement possible d'implémenter sur une machine adressable des procédures logicielles associatives. La solution matérielle ne se justifiera que si elle offre un gain notable de performances par rapport à la solution logicielle. La différence essentielle tiendra alors au temps de traitement. Autrement dit, ce n'est que dans le cas où un problème de type associatif n'est pas pris en charge de façon satisfaisante par les machines classiques que la solution câblée sera envisagée. Cette remarque est d'autant plus importante que, actuellement en tout cas, les technologies (cf I.3.2) permettant l'implémentation efficace de mécanismes associatifs restent relativement coûteuses.

I-2.3. Le problème des réponses multiples

On a vu qu'en considérant le système associatif avec un recul suffisant, on peut toujours admettre qu'une recherche associative porte globalement sur l'ensemble des données. On peut donc s'attendre à la non-unicité voire à la **multiplicité des cellules qui vont satisfaire au critère d'interrogation.**

Ce problème est bien spécifique aux mémoires associatives : dans le cas des mémoires adressables, le principe même de l'adressage repose sur la relation bi-univoque entre adresse et emplacement.

Le but de la recherche associative est précisément d'identifier un sous-ensemble des informations que contient la mémoire. Si les opérations qui doivent être effectuées sur ces informations peuvent l'être directement et localement, il n'apparaît évidemment aucune difficulté. Toutefois, dans la plupart des cas, les informations sélectionnées devront subir un transfert pour être traitées en un autre endroit du système. Il est alors nécessaire, d'une part, de connaître leur nombre, d'autre part, de déterminer un ordre de lecture.

I-2.3.1. *Comptage des réponses*

L'évaluation du nombre de cellules répondant au critère d'interrogation peut être plus ou moins précise, selon les besoins de chaque système.

Pour améliorer la simple indication binaire ("pas de" ou "des" réponses), SEEBER et LINDQUIST [39] ont proposé une valuation **ternaire** (pas de réponse/réponse unique/réponses multiples) dans un contexte de gestion de fichiers.

Dans le cas où le nombre exact de cellules activées s'avère nécessaire, il faut prévoir une fonction de comptage : elle revient à déterminer la somme de k nombres de 1 bit.

La plupart des solutions envisagées reposent sur des **structures arborescentes**. STOCKTON et FOSTER [40], par exemple, ont proposé un élément de base unique, l'additionneur "complet". Pour N cellules de stockage, il faut environ N circuits semblables et le résultat est élaboré en $\lceil \text{Log}_3 N \rceil$ étapes.*

Il est également possible de recourir à des techniques plus séquentielles. Dans ce cas, le plus souvent, la fonction de comptage sera associée aux procédés de sélection des cellules.

I-2.3.2. *Sélection de réponse(s)*

Considérons le registre A (activation) qui contient les bits positionnés correspondant aux cellules qui répondent au critère d'interrogation. Le problème est alors (ANDERSON [41]) de fournir à partir de ce vecteur A(j), un (ou des) vecteur(s) P(j) tel(s) que un seul des bits de P(j) soit actif. Le (ou les) vecteur(s) P(j) servira(ront) à repérer une cellule active, pour la lecture par exemple. Inversement, on peut envisager de calculer un vecteur I(j) (inhibition) tel que **toutes les cellules, sauf une** parmi celles qui ont été activées lors de l'interrogation, se trouvent inaccessibles à la lecture (figure 8).

* $\lceil x \rceil$ *dénote "le plus petit entier supérieur à" x*

Il serait fastidieux de décrire et comparer les diverses approches possibles (se reporter à la bibliographie).

Les solutions employées se situent dans un compromis entre **parallélisme** (et puissance matérielle) et **séquentialité** : on distinguera des procédures linéaires ou arborescentes.

a) Les solutions linéaires

Elles nécessitent peu de circuiterie additionnelle mais exigent un temps d'attente proportionnel au nombre de cellules. La plupart reposent sur la **scrutation** du registre A :

Le **décalage** bit-par-bit du registre de réponse permet de détecter la première cellule active (CHU [42]). Eventuellement, le comptage des décalages permet de générer une adresse pour la lecture. Des améliorations peuvent permettre, au prix d'un accroissement de complexité matérielle, d'accélérer le temps de sélection (voir l'exemple emprunté par FOSTER [43] à une étude Good Year).

Lorsque seule la détection du "premier" mot actif est souhaitée il est possible de mettre en oeuvre une technique de **propagation** automatique du signal de sélection. Ces techniques, également appelées "chaîne de bits", "cascade de bits", ou encore "ripple-carry", sont fonctionnellement comparables aux précédentes. Leur avantage est d'être intégrables au niveau des cellules de stockage.

b) Les solutions arborescentes

Au contraire des précédentes, elles nécessitent un matériel spécifique plus lourd mais leur temps de réponse ne croît que de façon logarithmique : elles deviennent donc plus intéressantes à mesure que N croît.

Les **réseaux arborescents** proposés sont, en fait, bidirectionnels : il faut propager l'activation depuis les feuilles vers la racine puis le signal de sélection (ou inhibition) en sens inverse. Ce double échange peut procéder de 2 phases distinctes (WEINSTEIN [44]) ou entrelacées, chaque noeud "dialoguant" avec son ascendant et ses descendants. Cette approche est d'ailleurs généralisable à d'autres fonctions (par exemple, la gestion des priorités) comme le montre ANDERSON [41] : la même structure

ADRESSE	A	I	P
0	0	0	0
1	0	0	0
2	0	0	0
⋮	⋮	⋮	⋮
j-1	0	0	0
j	**1**	**0**	**1**
j+1	1	1	0
j+2	0	1	0
⋮	X ⋮ X	1 ⋮ 1	0 ⋮ 0
N-1	X	1	0

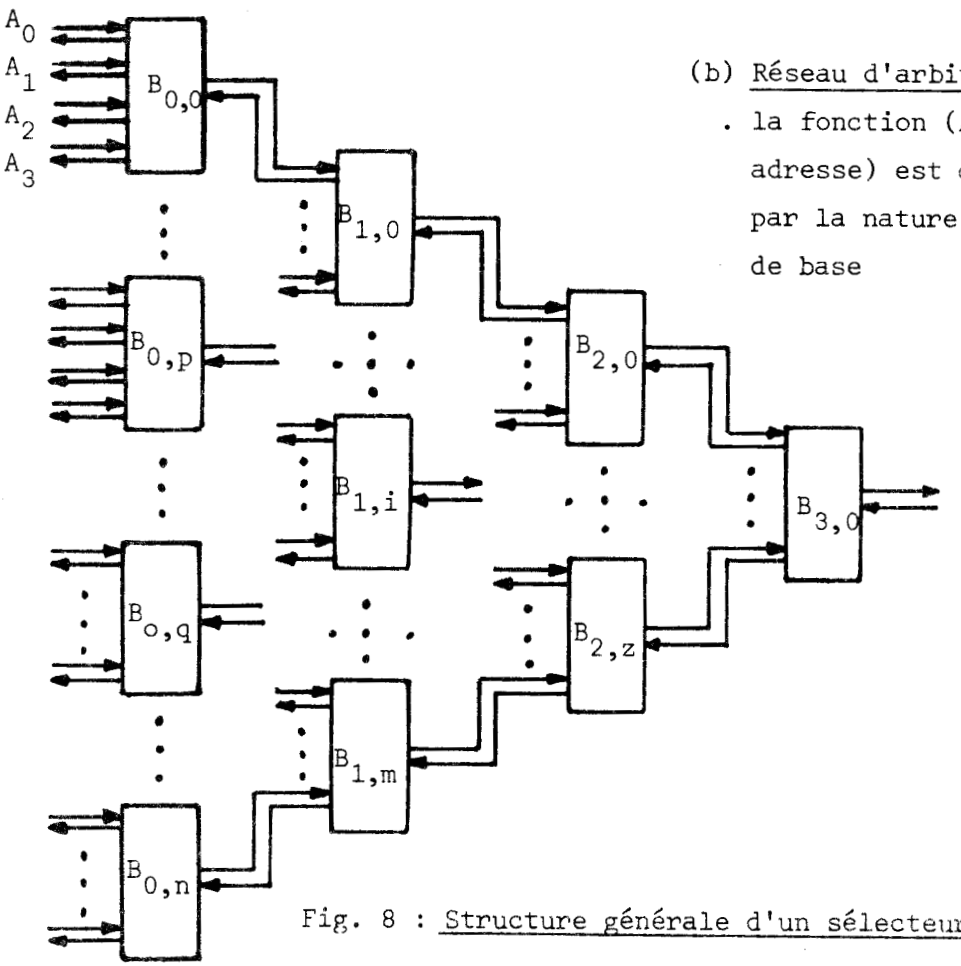
(a) Vecteurs de sélection

A : activation

I : inhibition

P : pointage

X = 0 ou 1



(b) Réseau d'arbitrage

. la fonction (A.I.P ou adresse) est déterminée par la nature du bloc de base

Fig. 8 : Structure générale d'un sélecteur arborescent

(d'après ANDERSON [41])

change de fonction suivant la fonction logique assumée par le bloc élémentaire (figure 8).

Cette approche présentant le défaut de se prêter moins bien à l'intégration (périodicité différente de celle des cellules de stockage) certaines variantes sont possibles : ainsi FOSTER [43] propose-t-il une structure déséquilibrée permettant d'économiser un certain nombre de portes, avec des performances comparables, et en obtenant une structure plus modulaire.

I-2.3.3. *Discussion*

La sélection des réponses est assurément la fonction la moins associative du processeur associatif. La lecture des informations à cet égard, représente un véritable goulot d'étranglement. Deux remèdes peuvent alors être suggérés :

- minimiser les transferts en traitant le plus possible l'information au niveau même du stockage (c'est un problème de matériel)
- affiner les critères et les algorithmes d'interrogation de façon à aboutir à une meilleure discrimination des réponses (c'est un problème de logiciel).

En fait, les méthodes de sélection évoquées ne répondent pas aux mêmes objectifs : les techniques arborescentes optimisent la lecture de la "première" réponse, ou réponse prioritaire. Les techniques purement séquentielles par contre, conviennent bien à une lecture exhaustive des réponses (cf. l'opposition accés aléatoire / accés séquentiel c'est-à-dire temps d'accès/débit dans les mémoires auxiliaires).

Du point de vue de l'implémentation, les deux types de méthodes s'opposent, tant au niveau complexité (nombre de portes) qu'à celui de la périodicité par rapport aux cellules de stockage. Ce dernier point peut s'avérer critique pour une intégration éventuelle des circuits.

Paradoxalement, certains auteurs semblent traiter séparément du choix de l'organisation associative ce problème de la sélection des réponses. Or il semble indispensable que ces deux aspects soient

considérés de manière interdépendante. En effet, le parallélisme obtenu au niveau de l'interrogation ne doit pas être pénalisé par une phase de sélection séquentielle ; et, inversement, il serait déraisonnable de doter une machine à organisation séquentielle d'un réseau câblé arborescent pour la sélection.

De même que la plupart des organisations de mémoires associatives reposent sur des compromis (I-226), il faut envisager une technique de sélection intermédiaire, compatible avec les caractéristiques du système au niveau du stockage. Un exemple intéressant est la solution proposée par CARLBERG [45], où l'on effectue une inhibition par propagation dans des blocs, un réseau arborescent permettant d'arbitrer entre les blocs.

I-2.3.4. *Remarque : Sélection des emplacements libres*

Outre le vecteur A d'activation, la plupart des auteurs (par exemple CARLBERG [45]) prévoient également pour chaque cellule un indicateur "emplacement libre" ou "information invalidée" ("tuée").

Ce vecteur, qui représente en fait l'état d'occupation ou de vacance des cellules de stockage, peut tout à fait être traité par un procédé analogue à ceux qu'on vient de décrire, voire par les mêmes circuits.

S'il s'agit seulement de trouver des cellules où l'on puisse écrire indépendamment des structures de données, le problème est alors résolu simplement : les techniques de sélection du "premier" emplacement libre seront alors efficaces.

Toutefois, dans le cas de structures de données élaborées, et/ou d'informations de longueur variable, la complexité sera tout autre : il semble que l'examen séquentiel des emplacements devienne alors préférable, associé à des réorganisations éventuelles du stockage. Les aspects propres à chaque implémentation interviendront et il est dès lors délicat de définir une méthode générale. En particulier, la manière de prendre en charge les structures de données (nous évoquons ce point au chapitre suivant) jouera un rôle important. Il s'agit alors de questions plus proches du logiciel.

Le concept même de mémoire associative entre ici en jeu dans la mesure où les emplacements ont une importance bien moindre que dans le cas des mémoires adressables : seul le contenu des cellules présente un intérêt, indépendamment des positions de stockage. Les difficultés apparaissent alors aux instants où il est nécessaire, physiquement, d'établir un lien entre l'espace des informations manipulées et les positions de stockage.

I-3 - ASPECTS MATÉRIELS ET TECHNOLOGIQUES

Les différentes architectures que nous avons passées en revue ne faisaient référence ni à des fonctions particulières, ni à des implémentations privilégiées. Cependant, on a vu que les problèmes matériels et technologiques interviennent dans une assez large mesure : en effet, les performances de ces machines sont directement liées au rapport entre la complexité des opérations disponibles (au niveau câblé) et le coût de leur mise en oeuvre. Rappelons que ces performances oscillent entre deux pôles qui sont, d'une part la capacité de stockage, d'autre part le temps de réponse.

Il est donc important de considérer quelles **opérations de base** le matériel prendra en charge, et quels **moyens physiques** peuvent rendre ces objectifs économiquement viables.

I-3.1. Opérations élémentaires

Lors de l'implémentation d'une mémoire associative, certains mécanismes de base sont nécessaires ; mais la notion même de "nécessité" apparaît toute relative et dépendra le plus souvent de l'application.

L'implémentation de toute fonction peut être envisagée dans une hiérarchie de choix reposant sur un équilibre variable entre matériel et logiciel. Le transfert d'une proportion de plus en plus importante des mécanismes vers le câblé concourt à une plus grande souplesse d'utilisation et à un accroissement des performances. La définition d'une frontière relèvera le plus souvent de considérations économiques.

Théoriquement, il serait possible de se contenter d'un dispositif logique de **détection d'égalité** avec masquage (figure 9) : on procède alors de façon générale en deux phases :

- interrogation : les cellules dont le contenu coïncide avec un champ donné (sélectionné par masquage) d'une clé de comparaison sont marquées (on dit aussi "activées").

Comparande	Information	Réponse
0	0	1
0	1	0
1	0	0
1	1	1

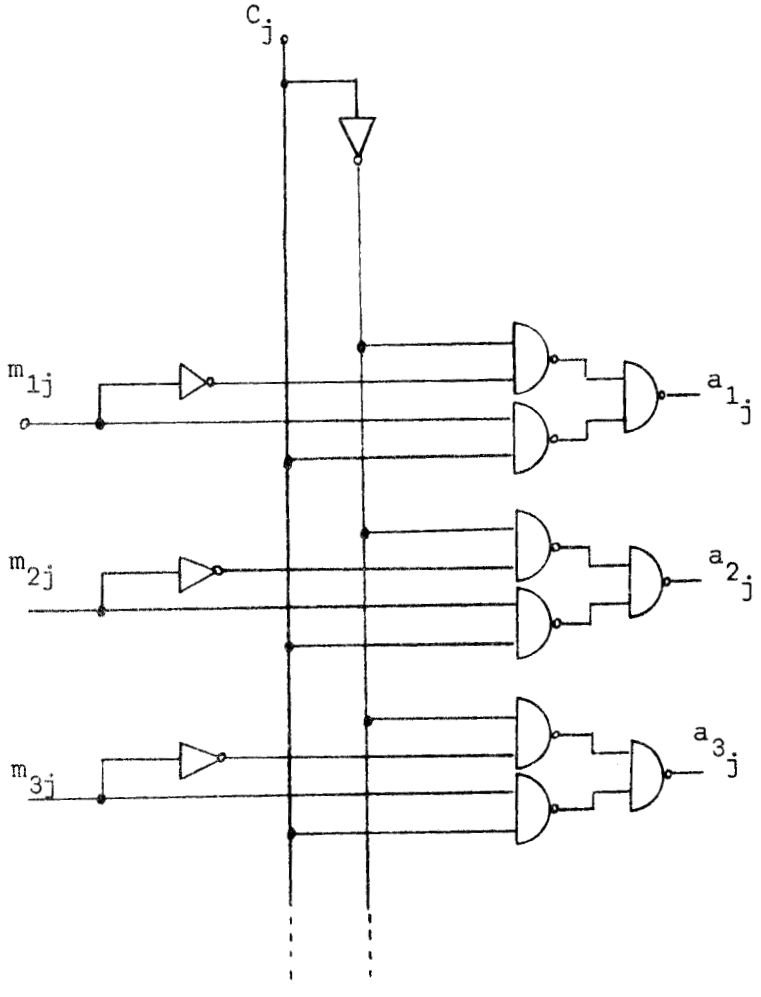
a) La fonction associative de base est le OU EXCLUSIF (ou son complément)

a_{i_j} = réponse fournie par la cellule m_{i_j}

m_{ij} = $j^{ème}$ bit du $i^{ème}$ mot stocké

C_j = bit j du comparande

b) Fonction d'interrogation associative (égalité) bit/bit d'une tranche de bit.



N.B. Il faut ensuite regrouper les résultats par mots (lorsqu'on interroge sur plusieurs bits/mots à la fois)

Fig. 9 : Les circuits d'interrogation élémentaire

- lecture/écriture : à la lecture, seules les cellules actives sont prises en compte. En écriture, on copie un champ donné de la clé dans le champ correspondant d'une (ou des) cellule(s) active(s). Dans ce dernier cas, on parle de "multi-écriture".

En principe **toute comparaison logique peut se décomposer** en interrogations élémentaires de ce type. Toutefois, dans bien des cas, cela s'avèrera fastidieux, et donc inefficace et insuffisant. On est donc amené à envisager l'implémentation de **primitives un peu plus compliquées** : celles-ci amélioreront les performances et faciliteront l'utilisation.

La plupart de ces primitives concernent l'**interrogation** et sont surtout applicables à des informations de type numérique. Ainsi, outre la détection égal/différent déjà citée, PARHAMI [2] énumère-t-il les critères suivants :

- inférieur/supérieur à (<, >)
- inférieur/supérieur ou égal à (≤, ≥)
- valeur maximale/minimale
- intérieur/extérieur à des bornes
- immédiatement inférieur/supérieur.

Notons qu'à la différence des autres primitives, la recherche du maximum (ou du minimum) se fait sans référence à un argument, à l'exception du masque : l'interrogation est alors du type "trouver l'enregistrement dont le champ démasqué est maximal (ou minimal)".

Dans la plupart des cas, seules les comparaisons simples (=, ≠, >, <, ≤, ≥) ont été câblées : la logique nécessaire est relativement peu compliquée et peut alors être associée à chaque cellule de stockage. De plus, il est possible de décomposer les opérations portant sur un intervalle, par exemple, en effectuant une séquence d'interrogations du genre ≥ ou ≤.

Les opérations plus globales supposent obligatoirement l'examen de tous les mots-mémoire - alors que, dans les comparaisons simples on peut trouver une réponse sans avoir achevé l'examen exhaustif de toute la mémoire -. Du point de vue pratique cela implique :

. soit une circuiterie logique coûteuse à 2 niveaux dans le cas du parallélisme : à la logique associée à chaque cellule doit s'ajouter une logique de concertation permettant le **regroupement des résultats** : à la limite, dans le cas de la recherche du maximum, cela conduirait à câbler un algorithme de tri.

. soit, dans le cas d'un fonctionnement séquentiel de type "daisy chain" un temps de recherche prohibitif.

En pratique, les recherches présentant un caractère plus global peuvent se décomposer en interrogations simples. Ces interrogations successives **restreignent au fur et à mesure le nombre de cellules examinées** si chacune d'elles porte sur les cellules activées lors de la précédente interrogation : pratiquement toutes les interrogations peuvent alors être simplifiées de cette façon quant au nombre de cellules interrogées.

Un autre point qu'il convient de considérer est le fait, déjà signalé, que ces critères sont essentiellement numériques : ils dépendent donc du **mode de représentation** de l'information (notammentles nombres négatifs). De plus, ils supposent une **relation d'ordre** qui ne sera pas forcément transposable à d'autres types d'informations,les textes par exemple. C'est bien à ce niveau qu'apparaît la difficulté : **dans le cas de la mémoire adressable, la fonction d'adressage reste la même quelle que soit l'information contenue** ; les possibilités de traitement sont en effet du ressort de l'U.A.L. **Par contre, la mémoire associative suppose un traitement effectif sur le contenu même de chaque cellule de stockage, quel que soit ce contenu.** A moins de supposer des moyens matériels assez considérables (pratiquement un processeur par cellule de stockage) il est indispensable de faire des **hypothèses** sur le type d'information traité, ce qui permet alors de privilégier une gamme d'applications. Le plus souvent, sans doute est-il difficile de faire autrement, les processeurs associatifs ont été axés soit sur le traitement numérique (STARAN, PEPE, ...) soit sur le traitement de textes (processeurs bases de données, ...).Une autre approche est de faire en sorte que les primitives câblées puissent convenir, grâce à leur caractère souple et général, à des types d'informations différents.

On a vu que les opérations d'interrogation constituent l'essentiel du fonctionnement du processeur associatif. Cependant, en tant qu'**organe de stockage**, celui-ci doit disposer également des opérations de **lecture** et d'**écriture**.

La **lecture** peut être envisagée de deux façons : directement ou par l'intermédiaire d'une adresse. Dans le cas de l'utilisation d'une adresse, celle-ci peut être produite par l'interrogation elle-même (un champ "nom" sert alors de réponse) ou par un circuit générateur d'adresse (voir I.2.3.2). L'avantage est alors d'avoir une mémoire "mixte", à la fois (ou tour à tour) associative et adressable. Cependant, il est plus intéressant, sur le plan de la rapidité, de se passer de l'adresse, en validant directement la lecture d'une cellule grâce à un signal produit par le sélecteur de réponse (cf I-2.3.2).

L'**écriture** peut également prendre deux formes : simplement, c'est une opération analogue à la lecture. Elle permet alors notamment de stocker une information dans un emplacement libre (cf. I.2.3.4). On peut également modifier tout ou partie du contenu d'une cellule.

Par contre, l'aspect parallèle (plus ou moins effectif) du fonctionnement donne la possibilité de la **"multi-écriture"** : celle-ci consiste à modifier un champ, choisi par masquage, dans un nombre quelconque de cellules qui auront pu être activées par une interrogation préalable. On peut ainsi évaluer une fonction booléenne à partir d'interrogations successives, ou même effectuer des opérations arithmétiques. Les propositions dans ce sens (voir notamment PORTER [46] qui donne des exemples) sont assez anciennes ; en effet, il s'agissait plutôt d'algorithmes ambitieux dont la réalisation matérielle reste assez complexe, compte tenu de l'état actuel de la technologie.

La définition d'algorithmes permettant de décomposer une opération arithmétique quelconque en opérations élémentaires disponibles de façon câblée relève plutôt du **logiciel** (et sera donc évoquée ultérieurement).

Là encore, les contraintes liées à l'application interviendront pour définir la part du câblé et du programmé dans la mise en oeuvre de ces algorithmes. La grande diversité des utilisations envisageables ne

permet donc pas de dégager de généralité à ce niveau. Cependant, si des opérations arithmétiques en un cycle sur tous les mots mémoire simultanément restent économiquement utopiques, des compromis peuvent être trouvés. On peut alors envisager l'implémentation de certains de ces algorithmes sur la base d'un traitement séquentiel par bits, en parallèle sur les mots, par exemple, comme dans les machines "bit-slice" (STARAN, ...).

I-3.2. Dispositifs technologiques

Notre propos n'est pas, ici, d'étudier dans le détail les principes physiques de fonctionnement des différentes technologies successivement mises en oeuvre dans la réalisation de mémoires associatives. En effet, la recherche de procédés physiques originaux [47] n'est généralement pas liée à un type d'application particulier, les mémoires associatives par exemple. Les progrés fondamentaux participent au développement de toutes sortes de dispositifs (circuits de traitement, mémoires, périphériques, ...). Les importants moyens nécessaires ne peuvent d'ailleurs se justifier que par cette diversité des retombées éventuelles. En ce qui concerne le traitement associatif, il est donc plus intéressant, en observant l'évolution de ces techniques, de tenter de discerner quelles voies peuvent apparaître comme prometteuses.

De nombreuses technologies ont été employées, proposées, ou envisagées pour implémenter le stockage associatif. Nous nous contenterons de les énumérer :

- **circuits cryogéniques** (superconductivité) [48] [49]. On peut noter un regain d'intérêt pour les très basses températures suscité par l'**effet Josephson** (voir l'effort qu'IBM consacre à ce domaine [50]). Le gain en rapidité considérable attendu de ces techniques serait précieux pour le traitement associatif.

- mémoires **magnétiques à éléments discrets** (tores, transfluxeurs, noyaux multi-axes, ...) [51] [52], à **films minces** [53] [54], ou à **propagation de domaines plans** [55].

- **mémoires optiques** [56] notamment par le biais de l'holographie [57] [58] et avec l'essor prochain des **disques optiques**.

- mémoires à **semi-conducteurs** [59], utilisées par exemple dans STARAN, PEPE, ECAM, ... , ou conçues comme un composant spécifique [60].

- **supports magnétiques auxiliaires**, ce qui correspond au domaine particulier des bases de données (voir I.2.2.5)

- **lignes à retard** en quartz ou verre [22] [23]

- **mémoires séquentielles électroniques**, enfin, qui semblent particulièrement intéressantes. Nous détaillerons plus spécialement, parmi les divers principes envisageables (voir, notamment, [61]), le cas des mémoires à **bulles magnétiques** (MBM).

I-3.2.1. *Choix d'une technologie*

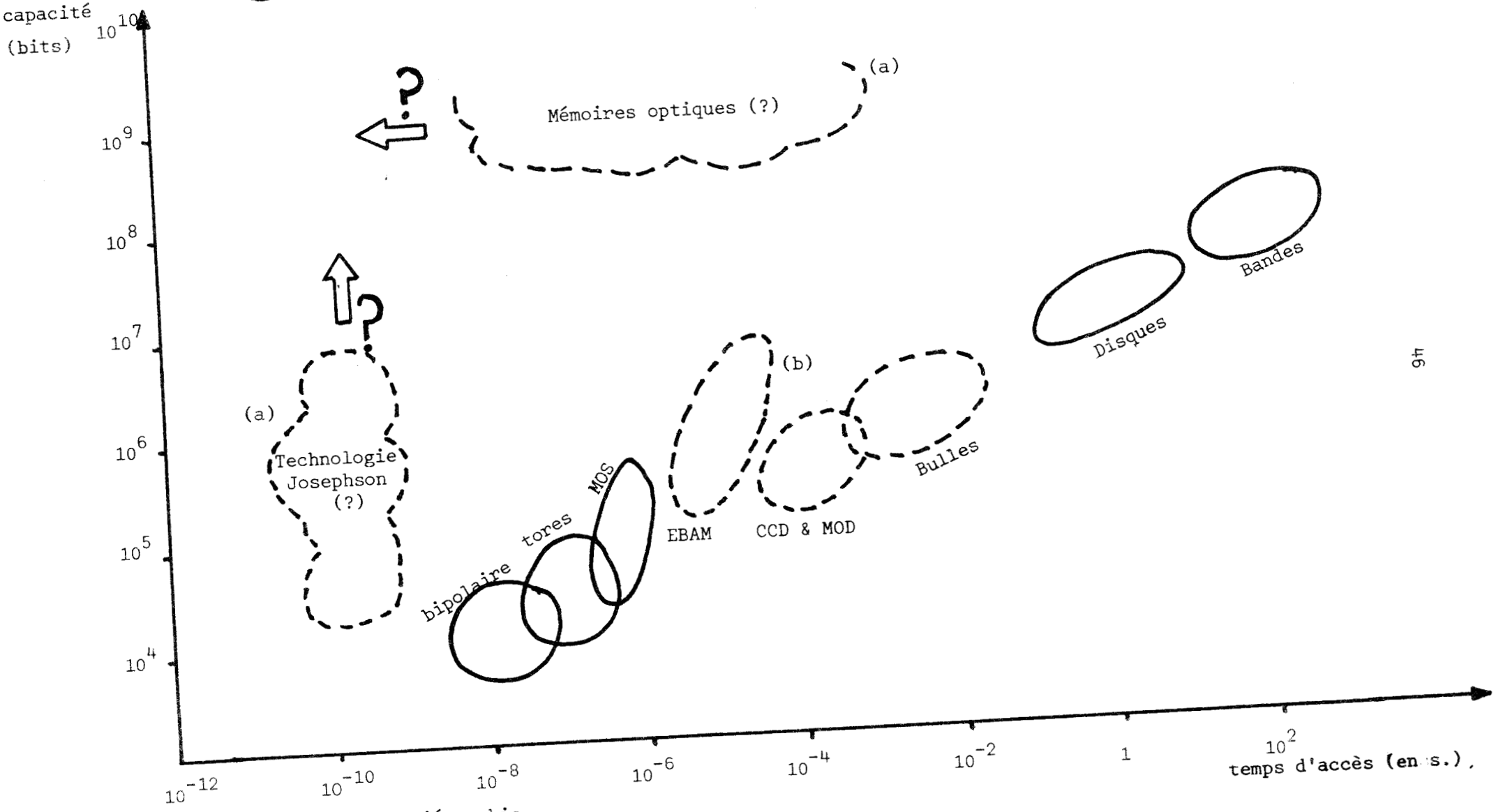
L'utilisation de l'une ou l'autre de ces technologies n'est évidemment pas arbitraire. Outre le fait d'être disponible et fiable, on peut dégager certaines caractéristiques liées à différentes interprétations du stockage associatif.

Pour le stockage classique, on a l'habitude de considérer la fonction mémoire comme hiérarchisée sur la base du rapport temps d'accès/capacité.

Or, dans le cas des mémoires associatives, la fonction ou la tâche spécifiques que l'on souhaite assumer imposent un niveau précis de performances, sans englober la totalité de la hiérarchie. Autrement dit, contrairement à la machine d'usage général qui peut nécessiter l'éventail complet des niveaux de la hiérarchie de mémoire, le sous-système associatif se fixe des "coordonnées" temps/capacité.

Pour simplifier, comme beaucoup de mécanismes associatifs supposent un examen exhaustif de la mémoire, on peut considérer que temps de réponse et capacité sont liés étroitement, de manière inversement proportionnelle. (Toutefois cela serait faux dans l'hypothèse d'une mémoire associative totalement parallèle).

Le critère déterminant est alors d'ordre économique et l'on pourrait le schématiser ainsi : **pour un temps de réponse** moyen (ou maximum) **donné, existe-t-il une technologie permettant, à un coût admissible, de construire une mémoire ayant la capacité voulue.**



a) extensions de la hiérarchie
b) Insertions dans la "lacune"

Fig. 10 : Apparition des nouvelles technologies

Or, dans l'éventail des technologies de stockage, deux tendances semblent s'affirmer (outre la progression régulière de l'ensemble des performances (voir figure 10) :

1) d'une part, des nouvelles techniques cherchent à **étendre la hiérarchie des performances aux extrémités**, en privilégiant :
- Le **temps d'accès** : à cet égard, compte-tenu de certains phénomènes physiques (tels que : énergie nécessaire à la distinction de 2 états, vitesse de propagation d'une onde électromagnétique, ...) le seul moyen de repousser les limites est le recours aux très basses températures (en particulier : effet Josephson).
- La **capacité** : l'enregistrement magnétique est limité par le "grain" des matériaux et les dimensions des entrefers. L'enregistrement optique, qui offre en théorie des densités énormes (> 10^8 bits/cm^2) semble appelé à prendre la relève.

2) D'autre part, la trou important au niveau du temps d'accès entre les mémoires mécaniques et électroniques tend à être comblé par des **techniques intermédiaires** telles que les bulles magnétiques (MBM), les registres CCD, voire les mémoires à faisceaux d'électrons (EBAM).

Ces deux approches s'opposent quant à l'importance des coûts par rapport aux objectifs. Dans le second cas, les mémoires associatives tendant à utiliser des architectures reposant sur des compromis séquentialité/parallélisme, le **stockage électronique séquentiel** semble une solution prometteuse.

Outre cet aspect décisif, certaines propriétés, liées aux caractéristiques physiques de chaque technologie, peuvent se révéler intéressantes dans un contexte d'application spécifique :

- **codage multi-niveaux** : les CCD, par exemple, bien que paraissant perdre du terrain permettent (voir GOSNEY [62]) de stocker en une même position de mémoire plusieurs bits d'information. A pouvoir d'intégration égal, cela peut multiplier par 2 ou 3 la capacité totale. Ce faisant, on introduit toutefois de nouvelles difficultés : détection plus délicate et moins bonne immunité au bruit (figure 11).

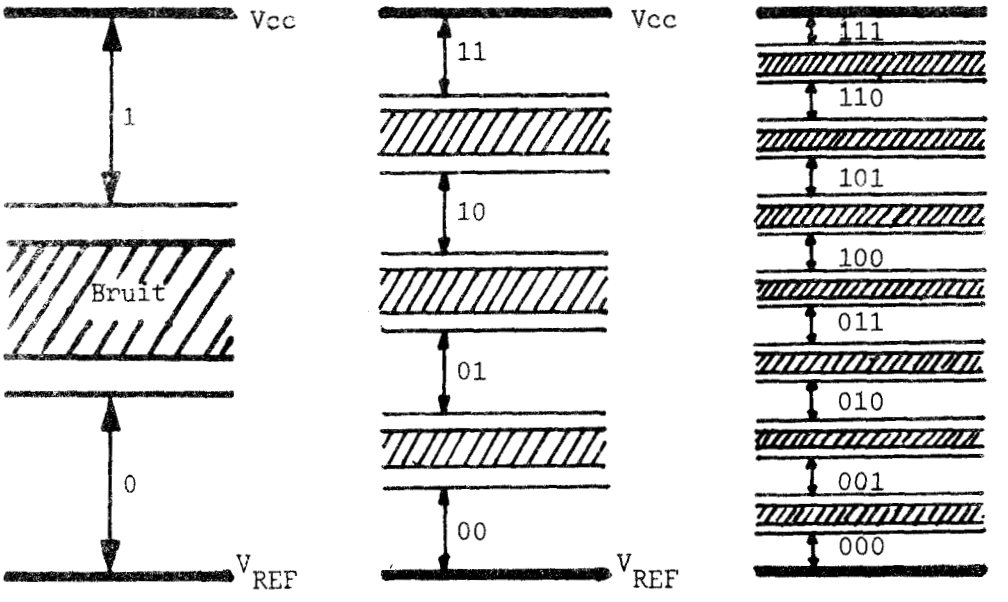


Fig. 11 : Accroissement de la capacité par codage multi-niveaux
(possible dans les CCD par exemple)

- **stockage analogique** : L'holographie, par exemple, peut offrir un moyen de mémorisation d'informations sous forme continue. Compte-tenu de certaines propriétés optiques (superposition, masquage, transformations géométriques, ...) cette possibilité pourrait se révéler fructueuse (cf. par exemple, la reconnaissance de formes, le traitement du signal, ...)

- **Compatibilité avec les circuits de traitement** : il peut être souhaitable d'homogénéiser les circuits de stockage et ceux de traitement : actuellement, ce sont surtout les semi-conducteurs qui présentent cette caractéristique.

- **non-volatilité du stockage** : bien que l'inconvénient de la volatilité puisse être minimisé (techniques de sauvegarde, alimentation-tampon, ...) un stockage permanent peut être nécessaire : les dispositifs magnétiques de toute sorte sont alors tout à fait indiqués, ainsi que certains dispositifs optiques.

- **non-inscriptibilité** : certaines applications reposent sur des informations définitivement stockées, sans mise à jour ultérieure (micro-programmes, dictionnaires, tables, ...). L'usage d'une mémoire morte (semi-conducteurs ROM, ou hologrammes, par exemple) permet un stockage plus fiable (conditions de conservation, immunité au bruit, ...) et un taux temps de réponse capacité plus faible.

- **possibilité d'accès aléatoire** : outre les accès associatifs il peut être souhaitable de disposer à certains moments, d'un accès aléatoire aux informations : dans ce cas, les technologies reposant sur une organisation séquentielle (registres électroniques, lignes à retard,...) sont à proscrire.

- **amovibilité du support** : des contraintes relatives à l'application peuvent aussi impliquer un stockage sur support amovible (notons qu'ils seront alors non-volatiles) ; outre les supports magnétiques classiques, on peut signaler les modules de mémoires à bulles magnétiques, ou de mémoires mortes.

Ces quelques critères ne constituent pas une énumération complète : seul un cahier des charges précis peut permettre de définir les caractéristiques que doit respecter la solution adoptée. Des contraintes supplémentaires pourraient alors concerner la tolérance aux pannes, les spécifications physiques (poids, volume, consommation, ...) etc ...

Un dernier critère qu'il convient de signaler réside dans la possibilité de doter une technologie donnée de la **logique d'interrogation** (même élémentaire) nécessaire aux fonctions associatives. A ce point de vue, les semi-conducteurs actuellement, sont certainement les plus adaptés, d'autant qu'ils permettent une intégration poussée. Par contre, certaines technologies de pointe (lignes à retard, optique, ...) nécessitent des dispositifs d'interrogation plus complexes. Toutefois, dans le cas où un tel coût est justifié, les possibilités de comparaison qu'on peut imaginer seraient beaucoup plus puissantes (par exemple, comparaison globale et directe de deux "images" holographiques).

I-3.2.2. *Des mémoires séquentielles électroniques : les MBM*

Il semble intéressant d'examiner plus attentivement le cas des mémoires à bulles magnétiques. Outre le fait que cela permet de voir de quelle manière la technologie peut s'adapter à certains besoins, un certain nombre de raisons justifient ce choix :

- les mémoires circulantes en général effectuent de façon automatique et implicite la **fonction de scrutation** : elles économisent à la fois de la densité et des voies d'accès.

- Les MBM, on l'a vu, représentent une **solution intermédiaire** entre le stockage lent à capacité importante et le stockage rapide à faible capacité. De ce fait, elles fournissent un support adapté aux nombreuses organisations associatives basées sur un tel compromis.

- Par rapport aux autres procédés analogues, notamment les registres CCD, les MBM semblent devoir prédominer comme en témoigne la double tendance actuelle :

. des constructeurs de composants (Intel, Rockwell, Texas Instruments, ...) à reporter leurs efforts depuis les CCD sur les MBM.

. des travaux nombreux suscités par les MBM dans des équipes de pointe (IBM notamment) dans le domaine technologique.

Ce double intérêt débouche, d'un côté sur la **commercialisation** de composants dont l'utilisateur doit définir l'usage, de l'autre, sur des **nouvelles propositions** de circuits basés sur les besoins en fonctions de pré-traitement.

- La non-volatilité et l'amovibilité du support (cf. la cassette amovible du constructeur japonais FUJITSU [63]), peuvent constituer des **avantages décisifs**.

Compte-tenu des remarques faites sur le choix d'une technologie, les caractéristiques des MBM les vouent à un rôle d'"anté-mémoire de masse", de mémoire de pré-traitement, disons de **mémoire de travail** : elles peuvent alors effectuer une fonction associative très spécialisée (mais très limitée en nombre d'informations) au niveau principal du système. A l'opposé, elles peuvent être décentralisées pour supporter des prétraitements tendant à minimiser les transferts superflus. Dans cette optique, on peut envisager d'y effectuer des tris (LIN [33]), des permutations (BONGIOVANNI [64]) des réarrangements (CHEN [65]) etc ... Toutes ces possibilités de manipulations des données pratiquement au niveau du stockage constituent dans un sens

des mécanismes associatifs. Cependant, elles nécessitent une organisation de mémoire spécifiquement définie, une logique additionnelle externe et surtout, elles privilégient souvent un type de données (entiers, textes,..)

La difficulté majeure des MBM réside dans l'organisation du stockage sur un chip. Des deux principaux types d'organisation (voir figure 12) c'est l'organisation série-parallèle-série (ou "boucle majeure-boucle mineure") qui semble devoir être préférée (en raison essentiellement des difficultés de fabrication). Or, même dans le cas d'une organisation SPS, l'accès au chip se fait séquentiellement (par la boucle majeure) et pénalise donc le temps de réponse (de l'ordre de la ms/mot).

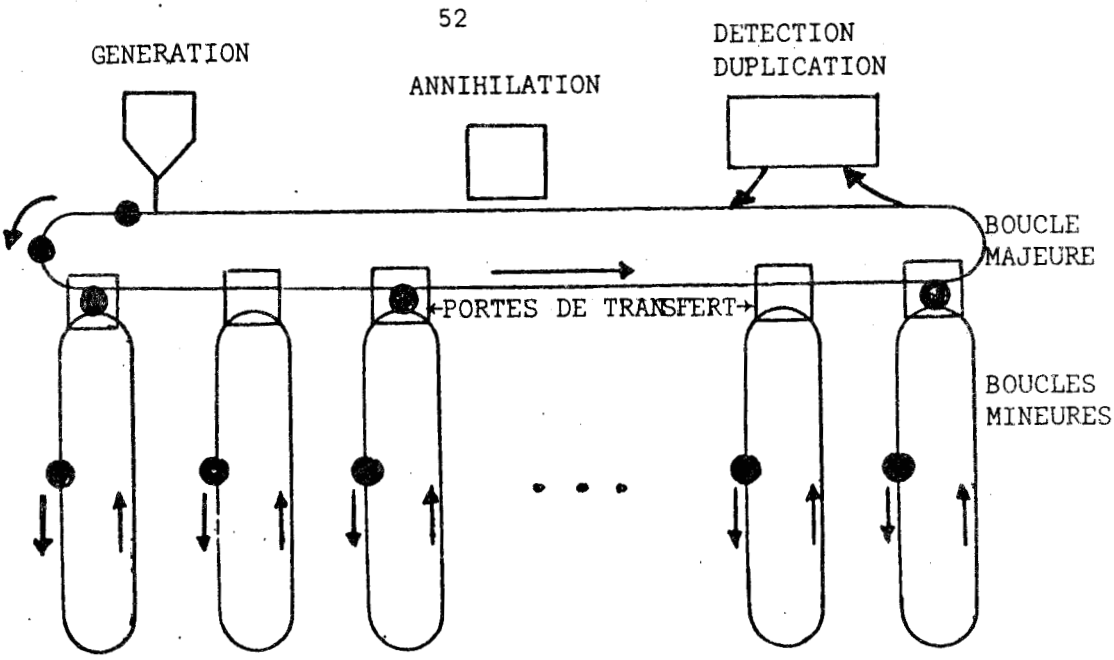
L'idée est alors d'incorporer, grâce à des motifs appropriés, des mécanismes logiques au niveau même des boucles mineures. Autrement dit, les portes de liaison majeure-mineure (figure 12a) deviennent programmables par un courant d'interrogation. Ainsi, mieux que la suggestion de TAKAHASHI [66] d'implémenter des fonctions logiques séquentielles pour effectuer de la reconnaissance de symboles, on peut doter les boucles de stockage de fenêtres d'interrogation internes au circuit.

Outre l'amélioration des performances par une plus grande décentralisation (en parallèle) de la logique d'interrogation, le fonctionnement sera plus général puisqu'on travaille au niveau du bit.

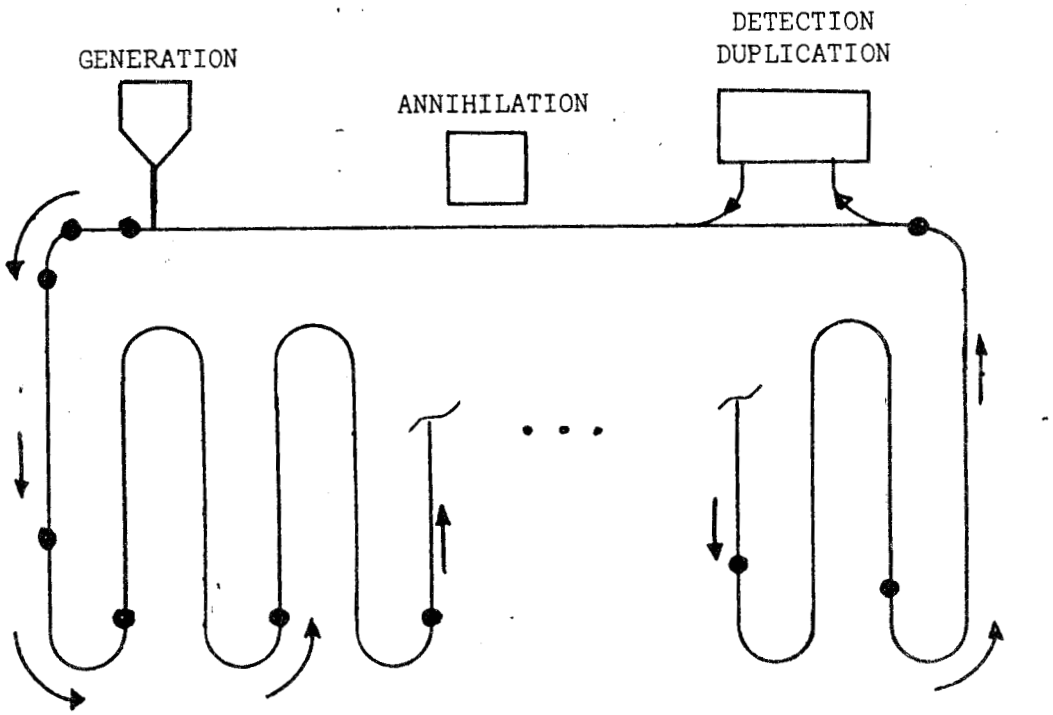
Le mécanisme proposé par LEE et CHANG [67] [68] va dans ce sens : chaque registre de stockage interne est doté d'un verrou programmable par le courant d'interrogation. Ils proposent également ainsi l'implémentation de réseaux logiques cellulaires adaptables au calcul de fonctions booléennes quelconques [67].

En fait, on peut observer que, dans son principe, ce genre de technique n'est autre qu'une implémentation originale des architectures associatives "bit-slice" du type STARAN. Cependant les MBM présentent alors des avantages non négligeables (coût, non-volatilité, ...).

Sur le plan technologique, signalons encore les mémoires à propagation de domaines (MOD) étudiées par la Société CROUZET,



a. Organisation SPS (série-parallèle-série)



b. Organisation série

Fig. 12 : Les deux principales organisations de MBM

Leur principe tient à la fois des MBM et des mémoires à film mince et elles présenteraient un certain nombre d'avantages [55] [69].

Enfin, toujours dans le cadre des organisations séquentielles, un rapport vitesse/capacité beaucoup plus élevé peut être atteint, si nécessaire au moyen de registres à décalage. Citons, par exemple, les registres FIFO de Monolithic Memories Inc. [70] : le boitier 67 402 A contient 64 mots de 5 bits circulant à 15 MHz.

Dans ce domaine particulier des mémoires séquentielles électroniques, on retrouve bien, finalement, tous les aspects relatifs au choix d'une technologie adaptée aux contraintes qu'impose l'application. Le fait positif est que le "continuum" technologique dont dispose le concepteur tend à être plus vaste : il permet donc une plus grande souplesse principalement en ce qui concerne l'adéquation du système aux besoins.

I-4 - ASPECTS LOGICIELS

De façon très marquée, le passage **des mécanismes associatifs matériels vers le logiciel a été négligé** –ce fait est notoire, et largement démontré par le volume relativement faible de travaux dans ce sens vis-à-vis de ce qui a été présenté plus haut –.

Plusieurs remarques peuvent étayer ce point de vue :

1) L'accès par le contenu, dans son sens le plus précis, est un **concept matériel** : les propositions d'architectures ou de technologies portent donc essentiellement sur une implémentation câblée de ces mécanismes.

2) De plus en plus, on est amené à considérer le traitement associatif comme une **activité spécialisée** : de ce fait, il est plus efficace de développer un logiciel d'application propre à chaque système : en effet, les systèmes opératoires, de plus en plus sophistiqués, se justifient par leur impact commercial, ce qui n'est pas encore le cas des machines associatives.

3) Le **coût plus élevé** – à l'heure actuelle – du matériel associatif oblige, lorsque la rapidité de traitement n'est pas un critère primordial, à travailler plutôt par simulation sur une machine classique. Dans ce cas, la plupart du temps, l'aspect associatif se trouve imbriqué dans un logiciel classique.

Notons d'ailleurs qu'une définition rigoureuse des performances recherchées permettrait le plus souvent de se satisfaire d'une implémentation sur une machine classique en utilisant le 'hash-coding' par exemple.

On peut en fait modifier l'affirmation faite au début de ce paragraphe de la façon suivante, si l'on distingue les trois domaines suivants :

1. algorithmes
2. programmation
3. simulation.

En effet, compte-tenu des remarques que nous avons faites, on peut considérer que :

- dans les quelques cas où l'implémentation câblée a été possible et effective, des algorithmes adaptés, puis un langage ont pu être proposés.
- Dans la plupart des cas, les propositions n'ont pas débouché sur des réalisations : le problème pratique de l'utilisation n'a donc souvent pas été abordé.
- Enfin, parfois, il a été jugé préférable de tourner la difficulté en recourant à la simulation.

I-4.1. Algorithmes

Une fois une architecture et une implémentation matérielles définies, on dispose d'un certain nombre de **primitives câblées.**

La différence essentielle par rapport aux machines classiques — où celles-ci sont pratiquement standardisées (transferts, opérations logiques ...) — est que, on l'a vu, il n'y a aucune généralité dans ce domaine. La fonction minimale indispensable est l'interrogation par comparaison d'égalité ; au-delà une grande variété de possibilités existe.

Les opérations associatives sont alors toutes décomposables en primitives de ce type. Ainsi FALKOFF [71] proposait toute une série d'algorithmes associatifs : cela reposait évidemment sur un hypothétique modèle de mémoire associative sans cependant supposer quoi que ce soit de l'implémentation. C'est une démarche quelque peu gratuite puisque, supposant telle ou telle primitive, les fonctions associatives se construisent assez naturellement. Cependant, il s'agit d'un aperçu assez intéressant des possibilités de mécanismes associatifs, dans la mesure où les hypothèses sont suffisamment générales.

L'associatif concerne l'accès et moins le traitement proprement dit ; à ce titre, il ne peut couvrir qu'une part des besoins d'une application.

On a, pratiquement, **deux types de démarches :**

- La démarche la plus courante, ascendante, est de bâtir les algorithmes sur la base d'un système câblé bien défini. Ainsi UNGARO [72],

par exemple, décrit des algorithmes de traitement numérique, de traitement d'images ...
Comme la plupart des systèmes réalisés effectivement n'ont pu l'être que par référence à un type d'application, les possibilités câblées sont assez variées. En conséquence, l'algorithmique nécessaire à l'emploi de ces machines peut difficilement se définir d'un point de vue général. Les opérations disponibles ne seront en fait que des **macro-instructions** typiques dépendant à la fois :
. des **primitives câblées disponibles**
. du **type des données traitées.**

- A l'inverse, on peut envisager les algorithmes à partir des structures de données, sans que l'implémentation intervienne. Cette stratégie a l'avantage de pouvoir être utilisée sur le matériel existant (adressable) en attendant de disposer d'un matériel associatif efficace.

Une présentation générale sur l'accès aux mémoires a été proposée par KNUTH [73]. Le contexte envisagé le plus souvent est plus restrictif puisqu'il s'agit d'**interrogation associative d'un fichier aléatoire.** Cela suppose en effet un accès par clés multiples à des enregistrements, sans se préoccuper de les traiter, ni le plus fréquemment de les combiner. Le principe est d'optimiser le nombre d'accès physiques par rapport à une requête donnée en employant :

- des algorithmes arborescents (voir par exemple BENTLEY [74])
- des techniques de hash-code (voir par exemple GOBLE [75]).

Des techniques intermédiaires peuvent être envisagées pour améliorer les réponses aux interrogations du type "plus proche voisin" (BURKHARD [76]). RIVEST [77] présente une évolution des performances de certaines de ces techniques ainsi que des possibilités d'améliorations (hash-code par blocs notamment).

Insistons sur le fait que ces algorithmes sont limités par la nature des possibilités d'interrogation. Les clés d'accès sont prédéfinies en liaison avec les algorithmes qui les utilisent parfois avec une structure de fichier particulière : c'est particulièrement nécessaire pour pouvoir choisir les fonctions de hash-code et déterminer la taille des blocs d'informations. De ce fait, on a affaire à des algorithmes relativement spécialisés, mais néanmoins efficaces pour tout ce qui relève de la gestion des fichiers.

I-4.2. Programmation

Deux approches doivent être envisagées :

- Programmation machine : la combinaison séquentielle d'instructions (éventuellement microprogrammées) permet la construction, dans un sens assez restrictif, d'un programme.

- Programmation évoluée : la démarche est opposée et vise à définir un langage de haut niveau exprimant les mécanismes associatifs.

Dans le premier cas, on risque de déboucher sur une utilisation pénible et peu souple. Dans le second cas, le problème de la compilation se pose dans la mesure où, on l'a déja noté, la mémoire associative convient mal à l'implantation de logiciel. Il faut se tourner vers la compilation croisée ce qui est souvent le cas pour les machines spécialisées.

Nous verrons que, dans la plupart des cas, le processeur associatif n'est pas une machine autonome mais que ce sera plutôt, puisqu'il convient à des tâches particulières, une ressource dans un système.

L'utilisation s'adressera le plus souvent à une machine-hôte. celle-ci, après compilation, enverra des commandes au système associatif de façon pratiquement analogue aux opérations d'entrées/sorties.

Les 2 aspects envisagés sont alors présents mais physiquement séparés :

. Le système associatif est commandable par un jeu d'instructions individuelles.

. La machine-hôte peut supporter l'interface utilisateur et en particulier un langage de programmation.

Les langages de programmation spécifiquement définis pour le traitement associatif sont rares : on peut citer le langage de haut niveau décrit par LANGE [78] pour STARAN. Le plus souvent, comme la compilation se fait sur une machine classique, on se contente d'incorporer à un langage évolué des sous-ensembles associatifs - c'est d'ailleurs la même stratégie que celle utilisée pour la simulation : à la limite, les langages "associatifs" de simulation qui ont été proposés pourraient après compilation

être interprétés pour adresser des commandes à un système associatif -

On peut également signaler le cas des bases de données relationnelles associatives. Les systèmes proposés prévoient soit un raccordement à une machine classique soit un fonctionnement autonome grâce à un langage spécialisé d'interrogation : l'exemple typique est celui de CASDAL, langage d'interrogation du système CASSM [25]

En ce qui concerne la programmation, deux types de caractéristiques s'opposent :

- le traitement associatif impose des habitudes nouvelles : possibilité de **parallélisation des tâches, aspect global des manipulations.**
- à ce prix, il libère l'utilisateur des contraintes liées à l'adressage et au déroulement séquentiel des tâches d'interrogation (boucles) d'où **simplification.**

Si les langages de programmation exécutables sur les machines classiques foisonnent, il n'en est pas de même pour ce qu'on pourrait appeler un "langage associatif" (mais en a-t-on vraiment besoin ?) pour deux motifs essentiels, qui dérivent tous deux du fait que le traitement associatif est fortement lié à la sémantique des données :

- relation entre algorithmes et données
- relation entre opérateurs câblés et données.

I-4.3. Simulation

Deux raisons essentielles peuvent justifier l'intérêt d'une simulation d'un processeur associatif :

- **vérification** d'un projet d'architecture matérielle et/ou éventuellement mise au point de programmes pour la future machine (il s'agit alors plutôt d'émulation)
- **remplacement** d'un dispositif câblé dont le coût aurait été excessif ou dont les performances souhaitées ne justifient pas un investissement particulier.

Dans le second cas, on transforme, par la définition d'un langage adéquat, un processeur classique en "machine associative virtuelle" : plutôt que d'utiliser des algorithmes séquentiels, il est préférable, alors, de recourir à des techniques de hash-code (voir I-4.2).

Plusieurs propositions ont été faites dans ce sens. Le plus célèbre de ces langages de simulation est sans doute LEAP, langage de type ALGOL défini par FELDMAN et ROVNER [79].

L'objet de base manipulé par LEAP est un triplet (A, O, V). Le triplet représente une **relation** A(O) = V c'est-à-dire :
l'Attribut A de l'Objet O est la Valeur V
exemple : (fils, André, Pierre) signifie
"le **fils** d'**André** est **Pierre**"
Sur cette base, les interrogations élémentaires consistent à masquer un ou plusieurs éléments des triplets :
par exemple : (A, O, ?) signifie, dans l'exemple
"Quels sont les **fils** d'**André** ?"
de même : (?, O, V) etc ...
Des combinaisons de ces interrogations sont possibles.

Sur ce modèle, d'autres langages ont été définis.
Citons par exemple :

- LAMBDA (LAngage de Manipulation d'Une Base de Données Associative) extension de CPL1, version conversationnelle de PL1 qui s'inscrit dans un contexte de terminal graphique (BOULLIER [80]).

- AMPPL II (Associative Memory Parallel Processing Language) construit comme une extension d'un sous-langage de traitement de listes de FORTRAN (FINDLER [81]).

- ASTROL, développé à partir de PASCAL (WIRTH [82]).

Le point commun à ces différents langages est de **manipuler une entité relationnelle de base.** On peut considérer que le modèle de bases de données relationnelles et ses développements (que nous avons évoqués en I-2.2.5) constituent une **généralisation de cette idée.**

D'autre part, le fait d'incorporer la simulation de mécanismes associatifs à des langages évolués très répandus peut sans doute se justifier par :

- l'expérience acquise dans la manipulation de ces langages qui facilite leur utilisation
- leur caractère suffisamment standardisé pour pouvoir envisager une compilation et une exécution sur des machines presque indifférentes.

I-4.4. Une mémoire associative câblée ou logicielle ?

Dans la mesure où, on l'a vu, l'implémentation câblée d'une mémoire associative pose des problèmes liés au coût et/ou aux performances, la simulation logicielle peut s'avérer une solution intéressante.

STILLMAN et BERRA [83] ont comparé, dans le contexte particulier de la graphique, les deux types de solutions. Les trois points de vue examinés sont les suivants :

- vitesse de traitement
- espace mémoire
- souplesse d'utilisation.

Il semble qu'en fait c'est surtout la **rapidité du traitement qui représente un avantage déterminant en faveur de la solution câblée.** Ce fait est d'autant plus vrai que la question d'accès est complexe, car la simulation nécessitera une recherche séquentielle plus fastidieuse et des intersections logiques de listes.

En ce qui concerne l'**espace mémoire**, la simulation semble plus exigeante. Cependant, il est évident que la comparaison est faussée puisque les deux solutions sont de nature différente : dans la solution câblée, on dispose d'une machine et d'un espace mémoire spécialisés alors que dans la simulation on a recours à une machine "banalisée" pouvant convenir à d'autres tâches.

C'est d'ailleurs dans cette optique qu'on peut considérer la mémoire associative simulée par logiciel comme étant plus souple qu'une machine câblée dont on fige les mécanismes.

Bien qu'il soit délicat de tirer des conclusions à partir d'une évaluation portant sur une tâche précise, le choix de la solution est assez clairement posé : c'est essentiellement la **vitesse du traitement** qui justifie l'implémentation câblée d'une mémoire associative. C'est-à-dire qu'il est nécessaire d'estimer finement cette contrainte par rapport au coût plus conséquent qu'elle entraîne. En outre, les solutions logicielles permettent une mise en oeuvre plus souple tout en exploitant l'expérience d'une programmation classique.

Il a été précisé, au début du chapitre, que la notion de "traitement associatif" pouvait recouvrir différentes réalités. C'est tout à fait évident lorsqu'on considère les développements parallèles, mais **presque indépendants**, des mémoires associatives câblées et des algorithmes associatifs par exemple. Dans le premier cas, on s'intéresse à l'accès physique au contenu d'une case mémoire ; dans le second il s'agit, par exemple, d'interrogation de fichiers par clés multiples. Les objectifs ne sont d'ailleurs pas comparables, puisque d'une part les progrès technologiques sont indispensables pour accroître l'efficacité des dispositifs câblés, d'autre part le logiciel utilise et se satisfait de la technologie existante.

I-5 - EXPLOITATION ET UTILISATION

I-5.1. Aspect spécialisé du traitement associatif

Il a été noté, au début de ce chapitre, combien le terme "associatif" pouvait être diversement interprété. Cela expliquait la multiplicité des façons d'appréhender cette notion (voir figure 1) et s'est vu confirmé par l'étendue des aspects qui ont été abordés.

Il apparaît que le caractère plurivalent du concept associatif convient d'être précisé en le confrontant aux deux notions duales d'adressage et d'accès par le contenu.

I-5.1.1. *Traitement associatif et accès par le contenu*

Beaucoup d'auteurs, et nous avons jusqu'à présent suivi cette tendance, considèrent ces deux notions comme indissociables. En fait, il n'est pas inutile de pouvoir les distinguer.

En effet, les mémoires accessibles (ou adressables) par le contenu relèvent précisément d'un mécanisme physique consistant à identifier une cellule de mémoire par l'énoncé de tout ou partie de leur contenu. A cet égard, le mécanisme dual est celui, classique, de l'adressage, et plus précisément de l'accés aléatoire.

Par contre, les mécanismes associatifs expriment des relations entre les données et permettent notamment, en exploitant ces relations, de retrouver une information, ou d'effectuer des traitements.

Si les algorithmes associatifs peuvent utiliser les mémoires accessibles par le contenu, cela n'a aucune espèce de nécessité. Il est cependant souvent admis que cette approche permet d'espérer des performances accrues, principalement en ce qui concerne la vitesse de traitement.

Comme le montre la variété des propositions "associatives",des lignes à retard aux algorithmes de hash-code, cette possibilité a été assez peu exploitée. On a plutôt assisté à un développement indépendant des techniques matérielles et logicielles.

La conséquence essentielle de cette distinction réside dans la possibilité d'effectuer du traitement associatif sans recourir à des techniques matérielles spécifiques.

Toutefois, par commodité et dans tous les cas où les deux notions sont confondues, les différents auteurs utilisent généralement le terme "associatif". (cf. I.1)

I-5.1.2. *Traitement associatif et adressage*

Outre le fait, que l'on vient de noter, qu'il est possible d'effectuer du traitement associatif sur une machine adressable, il est intéressant de pouvoir situer respectivement ces deux notions.

Le point de vue, qui pouvait sembler justifié a priori, qui tend à opposer symétriquement "associatif" à "adressable" doit être modifié. Si les différentes implications de cette approche (voir figure 1) restent valides, les deux notions ne sont pas à mettre sur le même plan.

. Une machine classique, adressable, regroupe un ensemble de niveaux relativement indépendants (matériel, logique, utilisation, système opératoire, ...). Autrement dit, la conception d'un système "universel" par définition est une fin en soi : le système doit être suffisamment souple pour servir d'outil dans des contextes aussi variés que possible.

. Au contraire, un système associatif repose de façon importante sur la nature des données à traiter, c'est-à-dire, en définitive sur le type d'application envisagé. De ce fait, le système associatif n'est pas un but, mais un moyen privilégiant une tâche précise. La démarche qui est alors le plus souvent suivie est de partir du problème posé pour définir les caractéristiques du système, notamment les primitives câblées.

En fait, s'il existe un archétype de monoprocesseur classique (machine de Von Neumann) indépendamment de l'utilisation qu'on en fait, il n'y a pas un mais des processeurs associatifs, chacun étant conçu dans l'optique d'une tâche précise.

La différence essentielle tient au fait suivant :

- La machine de Von Neumann est "dirigée par les instructions". De ce fait, elle dépend essentiellement de la structure des algorithmes qu'on réduit le plus souvent à une description séquentielle, et ce, indépendamment de la sémantique des données.

- Un système associatif est en partie "dirigé par les données" et en raison de la nécessité d'un traitement local de l'information (au niveau de la cellule de stockage), il privilégie obligatoirement une catégorie de problèmes.

Finalement, il convient donc de considérer le processeur associatif comme possédant un caractère spécifique. Cela impose, contrairement à un processeur "classique", d'envisager, d'une part la façon de l'insérer dans un ensemble de traitement de l'information, d'autre part les applications pour lesquelles il se révèlera indispensable notamment vis-à-vis d'une machine classique.

I-5.2. Environnement de la machine associative

On l'a vu, en aucun cas le processeur associatif ne peut se substituer à un processeur "classique" d'usage général. De ce fait, toute machine associative (processeur ou mémoire) doit être vue comme étant vouée à une fonction précise dans un ensemble de traitement. En conséquence c'est pratiquement la fonction qui lui est confiée qui conditionne son "statut", notamment en ce qui concerne ses relations avec l'utilisateur ou avec d'autres machines.

Les rôles qui peuvent être dévolus à une machine associative sont assez variés :

- périphérique spécialisé : le système associatif est connecté de façon analogue à une mémoire de masse (disque), par un canal par exemple. C'est typiquement le cas des processeurs relationnels de bases de données, souvent qualifiés de "backend".

- fonction de contrôle : on intègre à une machine un sous-système spécialisé dans une fonction donnée. Il s'agit alors d'un traitement associatif très limité par la taille mémoire nécessaire et par la spécialisation de l'opération prise en charge. L'exemple typique est celui des tables de conversion adresse virtuelle/adresse physique par mémoire associative qu'on trouve dans

certains systèmes (mémoire topographique). Des utilisations de ce type sont justifiées par la réduction sensible du temps d'adressage qu'elles permettent.

- contexte multiprocesseur : le processeur associatif est intégré à un ensemble, le plus souvent par liaison à une machine-hôte. La plupart des systèmes que nous avons décrits supposent cette relation (STARAN, ECAM, PEPE, ...). Notons que dans la plupart des cas le processeur associatif est considéré comme une ressource spécialisée. C'est particulièrement vrai pour PEPE dont le processeur de contrôle répartit les tâches séquentielles et parallèles entre une machine classique et le réseau associatif.

Notons que cette connexion du processeur associatif à un ensemble informatique est plus ou moins "serrée". Ainsi, CAFS, le processeur de gestion de fichier annoncé par ICL, peut être considéré comme une **machine autonome**. Cet exemple montre que les applications de gestion de données peuvent suivre une tendance opposée à celle des mémoires de masse "backend" : cette tendance pourrait consister à vouer la gestion d'une base de données à un système pratiquement autonome qui pourrait être :

- utilisé seul grâce à un langage interactif simple et puissant
- utilisé dans un contexte de réseau distribué.

Dans la plupart des cas de figure évoqués, la machine associative, en raison de ses possibilités spécifiques, a une position **passive**. En effet, son exploitation la considère comme une **ressource** ou au mieux comme un **processeur esclave**. C'est une situation analogue à un périphérique ou à un processeur arithmétique dans certains systèmes.

Il est donc nécessaire qu'une machine traditionnelle assure les fonctions de contrôle et d'E/S.

A ce niveau encore, apparaît le caractère spécifique du traitement associatif vis-à-vis des machines universelles. Loin d'être une sorte d'alternative ou de substitut des machines de Von Neumann, les processeurs associatifs ont un **rôle complémentaire** à assumer dans la conception des systèmes.

I-5.3. Domaines d'application

Les domaines d'application pour lesquels le traitement associatif a été cité comme un atout déterminant sont très nombreux. On trouvera une classification de ces applications dans HANLON [84] notamment.

Citons, entre autres, le traitement de signal, le traitement de texte, le contrôle aérien, le traitement d'images, la reconnaissance de formes, etc ...

Après l'enthousiasme initial qui a érigé le traitement associatif en une sorte de panacée, une caractéristique essentielle qui s'est imposée peu à peu a considérablement réduit l'intérêt : les processeurs associatifs, en dépit des services qu'ils peuvent offrir, ne peuvent en aucun cas, comme on l'a vu, convenir en tant que systèmes à usage général (voir HIGBIE [85]).

Autrement dit, l'énumération des applications relevant du traitement associatif est assez trompeuse : les différentes tâches envisagées par les auteurs peuvent en effet être du ressort d'un processeur associatif, mais pour chaque type de tâche, ce sera un processeur associatif **différent.** Cette remarque doit être cependant modérée si l'on considère des machines aussi puissantes que STARAN ou PEPE. Toutefois, il s'agit d'exceptions, en raison des coûts énormes induits par ces machines.

Au contraire, un processeur "classique" du type de ceux proposés sur le marché, est suffisamment souple pour convenir - parfois avec plus ou moins de bonheur - à des utilisations tout à fait diverses.

Un indice assez révélateur à ce point de vue est de considérer l'intérêt suscité par le traitement associatif chez les constructeurs :

- Il est (apparemment) assez faible chez les "grands" sur le plan commercial (comme IBM, Control Data, Burroughs, ...), pour qui l'aspect relativement standardisé du matériel classique est une préoccupation majeure. En effet, la stratégie est de toucher un marché le plus vaste possible, de façon à diffuser et à amortir le logiciel et la conception.

- Par contre, les machines de "haut de gamme" du côté associatif ont été proposées par des constructeurs plus "modestes" (du moins sur le

marché mondial) et ce, dans le cadre, le plus souvent d'un projet unique, spécialisé, mais coûteux. Citons STARAN (Goodyear Aerospace), RAP (Raytheon), OMEN (Sanders Assoc.), ALAP (Hughes Aircraft Co), PEPE (Bell Labs) etc ...

Deux cas peuvent sembler constituer l'exception à cette "règle" : ECAM (Honeywell) et le récent CAFS (ICL). Cependant, comme les précédents étaient des machines spécialisées dans le traitement numérique du signal, ces deux dernières machines sont à ranger dans la catégorie des processeuurs bases de données ce qui leur confère également un aspect très spécifique. Toutefois, sur un plan strictement économique, il semble que ces dernières aient un marché potentiel plus vaste.

Actuellement les deux domaines qui semblent vouloir recourir au traitement associatif sont donc :

- les **bases de données**
- le **traitement du signal en temps réel.**

Pour ce qui est des bases de données l'intérêt semble justifié essentiellement par la réduction importante des transferts d'informations à travers la hiérarchie de mémoire (voir I-2.2.5).

Par contre, l'intérêt des processeurs associatifs dans certaines applications relevant du traitement numérique est plus discutable. Les machines les plus connues ont été conçues dans le cadre du contrôle de trafic aérien ou du traitement de signal radar. Ainsi, décrit-on ce genre d'utilisation pour STARAN [86] [87] [88], etc ... Beaucoup d'algorithmes de traitement numérique ont été envisagés, que ce soit le filtrage temps réel, le traitement matriciel, etc ... (voir [89], qui contient de nombreux articles sur ce sujet).

En fait, bien que, réputées comme étant "associatives" des machines telles que STARAN, PEPE, etc ... **sont essentiellement des machines parallèles.** Cette qualité est particulièrement intéressante pour tout **traitement matriciel** où les besoins sont :

. rapidité globale du calcul : l'architecture SIMD permet de distribuer le calcul et donc d'accélérer l'obtention du résultat.

. couplage des données : en ce sens, l'interconnexion entre cellules de stockage et processeurs devient critique. Ce fait est

particulièrement mis en relief dans STARAN où une assez grande part des performances repose sur le réseau de commutation.

Ces types d'applications sont complexes surtout de par les algorithmes de calcul utilisés et le réarrangement rapide des données. On peut donc résumer leurs besoins matériels ainsi :

. puissance de calcul : d'où l'utilisation de techniques parallèles et/ou de processeurs spécialisés.

. rapidité d'interconnexion : d'où l'importance de l'intérêt actuel vers les réseaux d'interconnexion comme le montre une manifestation récente [90].

Ce n'est donc pas tant l'accès par le contenu que le parallélisme qui est le facteur déterminant des architectures proposées dans le cadre du traitement numérique.

Par contre, dans une autre gamme de tâches, des possibilités associatives peuvent jouer un rôle décisif : il s'agit d'applications telles que :

- reconnaissance d'écriture
- reconnaissence de parole
- traitement d'images

etc ...

On pourrait caractériser ces applications par l'une ou l'autre des approches suivantes :

- **reconnaissance de formes**
- **décodage d'une information bruitée.**

Dans ce cas, les algorithmes, du moins grossièrement, sont simples : il s'agit d'établir des comparaisons et/ou des corrélations entre une information inconnue et un certain nombre de références.

Par contre la difficulté du problème est due aux quantités d'information traitées et au caractère complexe des structures de données mises en jeu. De ce fait, l'accès global aux informations, et ce, par leur contenu, devient une nécessité, en particulier dès qu'il s'agit de pouvoir opérer en temps réel.

Quelques travaux dans ce domaine peuvent être signalés, notamment la reconnaissance de caractères manuscrits (YAU & YANG [91]). Toutefois, ces tentatives restent rares : en effet, la difficulté de la reconnaissance de formes est de définir les algorithmes et le codage optimaux. Or, toute machine associative, on l'a vu, implique des hypothèses à ce niveau. De ce fait, il est moins contraignant d'expérimenter sur un processeur "classique", tant que les algorithmes sont améliorables.

On a vu , en I.5.2, que des opérateurs associatifs pouvaient également assumer des fonctions très spécialisées, mais tout en étant un élément incorporé à un système classique. Un exemple intéressant, outre ceux déjà évoqués, est la sélection des travaux par un système opératoire (SMITH [92])

Finalement, contrairement au cas des processeurs classiques — dont la conception permet de ne se préoccuper de l'application que de façon secondaire — le traitement associatif, surtout lorsqu'il recourt au niveau câblé à l'accès par le contenu, repose sur une démarche inverse : c'est l'application qui, une fois définie, exprime un besoin en mécanismes associatifs. Ce besoin, selon le type d'application, se traduira ensuite par un usage plus ou moins partiel du traitement associatif.

Autrement dit, la conception d'un système associatif, à l'opposé d'un système classique, est plus un moyen qu'une fin en soi. De ce fait, l'approche doit en être totalement différente...

I-5.4. Conception d'un système associatif

Le panorama présenté jusqu'ici peut trouver deux justifications :

- essayer d'expliquer et de synthésiser le grand nombre de publications dans ce domaine.
- tirer des **enseignements utiles** pour la conception d'un tel système.

Si la première a un intérêt "académique" non négligeable, c'est surtout l'aspect "pragmatique" de la seconde qui semble profitable. Les

conséquences les plus marquantes que l'on peut en dégager portent d'abord, on l'a vu, sur la **démarche globale** du concepteur. Cette démarche doit tenir compte des différents problèmes propres au traitement associatif. Enfin il n'est pas inutile d'essayer de définir des aspects qui nécessiteraient des investigations plus poussées pour améliorer la souplesse d'utilisation de ces techniques.

I-5.4.1. *Principaux problèmes*

Nous nous intéressons ici à l'hypothèse d'une implémentation câblée d'un système associatif.

L'aspect matériel de cette étude nous a montré qu'actuellement le **coût matériel** est une difficulté majeure. Comme ce coût s'évalue surtout relativement aux solutions classiques, un système associatif câblé se justifiera essentiellement par le **gain en performances** obtenu par rapport à un système classique. Et dans ce domaine, il nous semble que seule la contrainte du **temps de réponse** représente un critère suffisamment pertinent. Cela, à condition de pouvoir définir avec précision le temps de réponse exigé par une application donnée.

Une des qualités d'un système associatif, est de **transférer les possibilités de traitement** — c'est-à-dire l'"intelligence" du système — **au niveau du stockage**. Les avantages principaux en sont de minimiser les transferts d'informations dans le système et les temps de traitement. Cependant, au niveau de la conception, cela pose un délicat dilemme, en ce qui concerne la **puissance des unités de traitement** ainsi distribuées : en effet, si ces unités de traitement, dans un but d'efficacité, ont des possibilités relativement évoluées, cela revient, outre un coût plus élevé, à les spécialiser. De ce fait, on privilégie inévitablement un type de tâches et donc on réduit la souplesse d'emploi du système. Inversement, si l'on choisit de garder une certaine souplesse, les unités de traitement (on dira plus volontiers la logique d'accès) resteront plus élémentaires et l'on risque d'obtenir des performances finalement assez modestes.

L'accès par le contenu, globalement à toutes les cellules, a, lui aussi, une conséquence défavorable : le problème de la **multiplicité**

des réponses (voir I.2.3.). Cela entraîne encore un surcroît de complexité matérielle. Mais, de plus, au niveau de l'utilisation, c'est-à-dire pratiquement de la programmation de ces systèmes, cette nouvelle caractéristique doit susciter de nouveaux modes de programmation. On peut remarquer que cet aspect a été assez peu abordé.

Enfin, les relations plus étroites entre les différents niveaux du système (matériel, algorithmes, programmation, utilisation, ...) si elles améliorent l'efficacité de l'ensemble, compliquent encore la conception : les contraintes sont plus rigoureuses compte-tenu de l'interdépendance accrue, et il est pratiquement impossible de traiter chaque niveau indépendamment.

I-5.4.2. *Discussion sur une démarche de conception*

L'approche la plus couramment utilisée dans la conception d'un système est ici caduque.

De même, contrairement à ce que l'on pouvait penser a priori, il ne semble pas qu'existent des applications privilégiées qui puissent trouver dans le traitement associatif une solution complète et exclusive. S'il en était ainsi, la démarche consisterait à :

- caractériser un contexte d'utilisation, ou mieux, définir les qualités demandées à une tâche pour tirer profit du traitement associatif. Cela supposerait donc que l'on définisse des classes d'algorithmes, des types de données, une stratégie de contrôle de l'exécution qui puissent convenir à ce genre de traitement. Notons que cela supposerait, par corollaire, l'existence d'un **modèle rigoureux et universel** des systèmes associatifs, ce qui est loin d'être le cas.

- envisager une étude de réalisation qui englobe simultanément l'implémentation matérielle, l'architecture du système, la programmation etc ... Cette organisation de l'étude serait complexe et lourde, tout au moins pour la plupart des structures de recherche susceptibles de s'intéresser à ce domaine.

Par contre, il est non moins évident qu'un certain nombre de problèmes, non résolus de façon satisfaisante par les solutions classiques ont potentiellement avantage à utiliser le traitement associatif.

Cependant cette utilisation doit considérer le traitement associatif comme un **outil** au même titre que des techniques d'accélération (anticipation, microprogrammation, ...) ou des architectures évoluées (multiprocesseurs, parallèles ou "pipe-line", machines synchronisées par les données, ...) Notons cependant que ces dernières ont l'avantage de pouvoir être développées "pro arte" dans la mesure où elles contribueront à coup sûr à l'amélioration générale des performances.

Finalement, le traitement associatif doit être considéré comme une possibilité parmi d'autres dans la recherche d'une solution globale à un problème donné. **C'est donc dans le cadre d'une stratégie globale de conception que le fonctionnement associatif trouve sa place et non en tant que système autonome.**

De ce fait, pour modifier le point de vue représenté a priori dans cette étude (figure 1), on peut dire qu'il n'existe pas de démarche associative indépendante, mais que dans un environnement plus global, cette composante peut apparaître. Ainsi on pourrait schématiquement représenter la genèse d'un système associatif de la façon suivante (figure 13).

-**identification (éventuelle) de tâches relevant de mécanismes associatifs** (voir I-5.3,)

- **évaluation du compromis coût/performances** permettant, le cas échéant, d'éviter un choix contestable d'une implémentation matérielle. HIGBIE [85] note, par exemple, que le caractère exotique d'une architecture ne justifie pas son utilisation dans le cadre de certains problèmes (le contrôle de trafic aérien par exemple).

- étude des **conséquences sur la programmation** induites par le traitement associatif. Cela peut impliquer la définition d'un langage de programmation ou des modification de la stratégie d'ensemble. En tout cas le point délicat est l'**interfaçage** avec le reste du système en raison de la multiplicité des aspects qu'il revêt.

I-5.4.3. *Lacunes actuelles du traitement associatif*

Outre le point de vue exprimé plus haut d'une approche différente de ces techniques, un certain nombre de domaines nécessitent un approfondissement, de façon à assurer un fonctionnement plus efficace et surtout une souplesse d'emploi accrue.

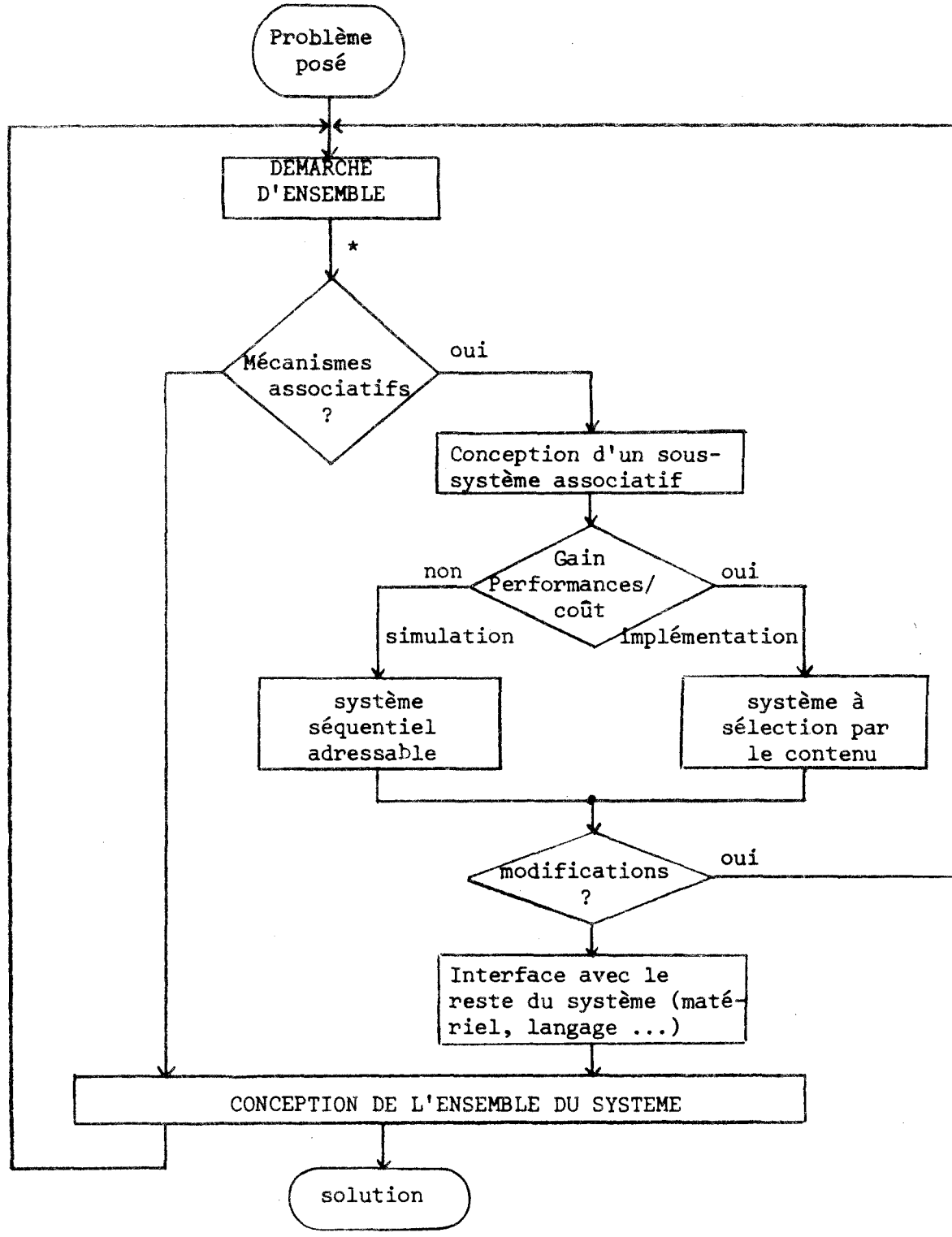


Fig. 13 : Intervention du traitement associatif dans la conception d'un système

* NB : On a détaillé le "sous ensemble associatif" éventuel sans imposer un caractère obligatoire ou prioritaire à cette première étape.

La **fiabilité et la tolérance aux pannes** de ces dispositifs ont été très peu explorées. Une hypothèse couramment répandue admet que les systèmes associatifs **tolèreraient** mieux un fonctionnement dégradé : cela demanderait vérification. Une façon d'éviter cette question est de distinguer les côtés logiciel et matériel. Au niveau du logiciel associatif, comme aucun processeur associatif ne gère son logiciel (rôle dévolu à la machine - hôte), le problème ne se pose pas. On peut cependant supposer que la plus grande simplicité d'emploi des primitives associatives réduit les risques d'erreurs. En ce qui concerne le matériel, l'aspect répétitif et parallèle (modulaire) du processeur associatif est contrebalancé par une plus grande complexité d'une cellule. Comme actuellement les techniques employées ne sont pas spécifiquement associatives, on rejoint les problèmes généraux de fiabilité et de tolérance aux pannes : il conviendrait d'extraire des approches existantes de ces problèmes ([93] [94]) des notions applicables au traitement associatif.

L'**évaluation des performances** est également un besoin impérieux. En effet, jusqu'ici les justifications sont basées sur des comparaisons avec les machines séquentielles principalement sous l'aspect temps de réponse. Cette approche, quoique utile, est partiale : en effet, une machine séquentielle peut rendre des services pour lesquels un processeur associatif s'avèrerait inefficace. Cette attitude est due en fait à la compétition entre processeurs associatifs et processeurs séquentiels. Or, on l'a vu, la complémentarité de ces 2 types de machines rend cette rivalité sans objet. Il reste néanmoins qu'il est utile de pouvoir connaître avec précision les gains en performances qu'apporte l'implémentation câblée d'une tâche. De même le concepteur doit pouvoir tenir compte de façon claire des compromis rapidité/coût qu'induisent les différentes organisations associatives disponibles.

Les **techniques logicielles nouvelles** que réclame l'introduction des mécanismes associatifs doivent être envisagées. En particulier, on suppose que l'aspect global des primitives d'interrogation facilitera la programmation de ces mécanismes, mais cela doit être vérifié. De plus, les avantages acquis grâce au traitement associatif ne doivent pas être payés par une complexité d'utilisation accrue. A cet égard, la définition des langages d'interrogation, voire des applications, doit tenir compte

de stratégies permettant d'optimiser les services offerts par le sous-système associatif. Une réflexion sur les **structures de données** peut être profitable (ainsi que le montrent **les modèles de bases de données**).
Sur la façon d'exploiter de façon optimale des possibilités associatives et ce, de manière quasi-naturelle, des rapprochements sont à effectuer dans le domaine
- des **langages d'interrogation** (ZLOOF [95], MESGUICH [96], COULON [97] par exemple)
- des **questionnaires** (CASEY [98], PICARD [99], ...).

Plus généralement, tout ce qui précède justifie largement la nécessité d'aborder ces problèmes avec plus de recul de façon à minimiser l'influence des habitudes issues de l'usage des processeurs séquentiels.

CHAPITRE II

DESCRIPTION FONCTIONNELLE D'UNE MEMOIRE ASSOCIATIVE

On a vu qu'un des principaux problèmes posés par la mise en oeuvre du traitement associatif réside dans son originalité même. En effet, le traitement associatif, en tant qu'outil spécialisé, ne peut s'accommoder des techniques d'utilisation habituelles. En particulier, cette remarque est essentielle pour préciser le rôle du stockage associatif par rapport aux mémoires adressables.

De même qu'une mémoire adressable n'est vue par l'utilisateur (au sens large : programme, processeur, opérateur, ...) que comme un ensemble d'emplacements, il importe de tenter de définir quelles sont les fonctions des dispositifs associatifs. Cela suppose de prendre un certain recul vis-à-vis de notions "classiques" devenues très (trop) usuelles.

Dans ce qui suit, on essaie donc de décrire, par rapport au "milieu extérieur", le fonctionnement de ce système que constitue la mémoire associative. L'approche adoptée — une modélisation rigoureuse pouvant s'avérer complexe et dépasser le cadre de cette étude — tend donc plutôt à être intuitive que formelle.

II-1 - PROBLÈME DE LA STRUCTURATION DES DONNÉES

La mémoire "évoluée" que l'on essaie de définir doit pouvoir s'adapter aux besoins de l'utilisateur.

Son aspect fonctionnel repose donc sur les possibilités de manipulation des objets qu'elle doit offrir.

Il est nécessaire de considérer les structures de données d'un point de vue général. Les spécifications de la mémoire à ce niveau pourraient aussi bien conduire à une implémentation en mémoire adressable. En effet, l'aspect associatif des manipulations d'objets est finalement indépendant de leur représentation puisqu'on a vu qu'il était possible de simuler des mécanismes associatifs sur une mémoire adressable à l'aide d'un logiciel approprié.

On verra cependant que dans les mémoires classiques cette indépendance entre niveau fonctionnel et représentation physique n'est pas toujours respectée, ce qui conduit à la subordination des possibilités logicielles au matériel ; le cas le plus probant est l'adéquation des structures de données employées classiquement aux notions d'adresse et d'emplacement.

Les multiples études des structures de données ([100] [101] [102] [103]) montrent qu'une approche fonctionnelle des informations doit rester indépendante de toute implémentation, du moins dans la mesure où celle-ci ne doit pas influer sur les possibilités d'utilisation. On rejoint ici des préoccupations des spécialistes des bases de données, notamment les modèles relationnels (CODD [26])ou autres (par exemple PIN & CHEN [104]).

Nous verrons, après avoir examiné le cas des mémoires adressables, comment prendre en compte les fonctions des structures de données dans une mémoire associative. La description fonctionnelle de la mémoire associative en découlera, en essayant de dégager du modèle les aspects qui seront les plus intéressants.

II-1.1. Approche fonctionnelle des informations

On distingue d'habitude deux types d'objets (PAIR & GAUDEL [103])

- des **objets élémentaires** (entiers, réels, booléens) : on notera dans le contexte associatif de l'étude que ce sont purement des valeurs.

- des **objets composés** (tableaux, fichiers, arbres, listes, ...) : ce sont des objets élémentaires que l'on a regroupé et organisé à l'aide d'une **structure**. Cette structure est caractérisée par les **fonctions de manipulation** des objets élémentaires qu'elle contient. Ici, outre les valeurs,on a des fonctions paramétrées qui traduisent l'ordre induit par la structure. L'usage veut que l'on appelle ces objets composés des structures de données.

Caractériser un ensemble d'informations consiste d'abord à mettre à la disposition de l'utilisateur un certain nombre de types élémentaires puis à définir les structures construites à l'aide de ces objets élémentaires. Pour définir les structures de données, il faut définir les fonctions qui sont attachées à ces structures. Or,il faut distinguer les fonctions proprement dites de la façon dont on les réalisera.

Il semble intéressant de distinguer 3 niveaux de description des structures de données (MEYER [105]) ; ces niveaux sont la spécification fonctionnelle, la description logique et la représentation physique.

- spécification fonctionnelle : elle comprend :

. un ensemble de fonctions qui peuvent être de trois sortes :

  création

  modification

  accès

. un ensemble de relations : ce sont des énoncés logiques que doivent vérifier les fonctions.

On ne connaît alors la structure que comme une boîte noire sur laquelle il est possible de faire certaines manipulations.

- description logique : c'est le procédé de construction des objets composés. On donne un certain nombre de types de base (vocabulaire) et l'on a des règles syntaxiques faisant intervenir des opérations simples (union, juxtaposition,concaténation, ...) de façon éventuellement récursive.

- représentation physique : c'est ici seulement que l'on se préoccupe de l'implémentation. Classiquement, les méthodes font appel à l'utilisation d'indices, de pointeurs, de tables, d'indicateurs, ...

Entre ces trois étapes, la distinction n'est pas toujours aussi nette, et notre approche, parce qu'elle concerne une implémentation différente des réalisations classiques, englobera les niveaux fonctionnel et logique.

La différence essentielle entre les 2 niveaux repose sur le fait que, du point de vue fonctionnel, on ne sait rien des objets que contient la structure. Au niveau logique, on précise quel(s) type(s) d'objet(s) contient la structure et la façon dont ils sont organisés.

L'exemple qui suit illustre ces définitions.

- Exemple ; au niveau fonctionnel, rien ne distingue une pile d'entiers d'une pile de caractères. La spécification fonctionnelle d'une pile serait la suivante (T est l'ensemble des objets du type constituant la pile):

. fonctions : "créer-pile" :→ PILE (création)
"vide" : PILE→ LOGIQUE (accès)
"dépiler" : PILE→ T x PILE (modification)
"empiler" : T x PILE→ PILE (modification)

. relations

$\forall$ t $\in$ T, $\forall$ p $\in$ PILE
vide (créer-pile)
T vide (empiler (p, t))
dépiler (empiler (t, p)) = [t, p]

Cette définition est simplifiée. Pour rendre compte exhaustivement du fonctionnement d'une pile, certaines précisions supplémentaires seraient nécessaires (cf. MEYER [105]).

La description logique, par exemple, pour une pile d'entiers, serait :

type PILE D'ENTIERS = VIDE | (sommet = ENTIER ; suite = PILE D'ENTIERS)
où ; note la justaposition
| note l'union (logique).

Décrire fonctionnellement les structures de données, c'est donc :

- préciser le(s) type(s) élémentaires(s) qui la compose(nt)
- décrire les fonctions de cette structure.

En outre, on a besoin de paramètres sans contenu significatif mais qui servent à l'accès ou à la création (exemple : taille d'une pile, indices d'un tableau, ...)

Le plus souvent, au niveau fonctionnel, les paramètres sont inutiles ou implicites. Ainsi les structures les plus courantes utilisent souvent des opérations de base telles que successeur, concaténation ...

Il conviendra donc de distinguer les cas où les paramètres ont un intérêt sémantique incontestable des cas où ils ne servent qu'à faciliter la description logique ou l'implémentation.

II-1.2. Cas d'une mémoire adressable

De ce qui précède, il ressort que classiquement on a pris l'habitude de décrire un objet composé (ou structure de données) à l'aide de 3 composantes :

- un ensemble de valeurs (ou objets) élémentaires
- un ensemble de fonctions de manipulation
- un ensemble de paramètres

Dans une mémoire adressable, lors de l'implémentation, les paramètres d'accès se traduisent en termes d'emplacements physiques (exemple : l'objet de rang n d'une liste est repéré en ajoutant k x n à l'adresse de début de liste). La relation entre l'emplacement physique et la fonction d'accès qui lui est associée est plus ou moins simple suivant le type de structure considéré et le mode de représentation choisi.

En outre, des raisons d'optimisation ou d'amélioration du fonctionnement ont conduit à l'utilisation de niveaux intermédiaires. Des mécanismes plus sophistiqués ont été introduits, comme par exemple, les mémoires topographiques ou les fonctions de hash-code. Cependant, dans tous les cas, on a bien une relation (biunivoque, du moins c'est souhaitable) entre l'emplacement physique d'un élément d'information et son

aspect fonctionnel (c'est-à-dire fonctions de manipulation + paramètres). On dit souvent que cette relation est une chaîne d'accès (cf. CROCUS [106]).

Du point de vue des structures de données elles-mêmes, il semble que l'on puisse en distinguer deux "familles" :

- d'une part, certaines peuvent être considérées comme "naturelles" en ce sens qu'elles sont liées à l'utilisation de façon sémantique : ce sont, par exemple, les tableaux pour le calcul matriciel.

- d'autres, au contraire, semblent "artificielles" : leur seule raison d'être est de structurer (ou plutôt d'ordonnancer, voire de hiérarchiser) un espace d'informations en vue d'optimiser son exploitation dans une machine classique (à adressage). Citons, par exemple, le fichier séquentiel, utilisé pour mémoriser un ensemble non structuré d'objets.

Dans ce cas, on peut penser que c'est le mécanisme (la contrainte ?) de l'adressage qui impose un rangement efficace des informations. Dans l'autre cas, s'il existe du point de vue de l'utilisateur une topologie (sémantique) à respecter c'est la notion d'adressage qui conduit à introduire un couplage étroit entre l'espace fonctionnel et l'espace physique. Ainsi, par exemple, si, d'un point de vue fonctionnel, l'article DUPONT se trouve "entre" les articles DUBOIS et DURAND, rien ne prédispose les emplacements qu'ils occupent à respecter cet ordre, sinon des considérations liées à l'implémentation.

Le mécanisme de l'adressage permet donc d'établir des correspondances simples entre les 3 niveaux de description des structures de données. Dans ce sens, on peut considérer qu'il favorise, dans une certaine mesure, la prise en charge de leur représentation.

II-1.3. Traitement associatif et structures de données

Dans le cas d'une mémoire associative, la notion d'emplacement physique n'est plus utile. Or, on a vu que, généralement, les représentations des structures de données reposent sur la topographie de l'espace mémoire. Dans une mémoire associative, en général — il peut être prudent de réserver ici la possibilité d'envisager ultérieurement des exceptions—, on ne cherchera pas à utiliser des relations entre l'emplacement physique d'un objet représenté dans l'espace mémoire, et son "emplacement" fonctionnel dans l'espace de données de l'utilisateur.

Autrement dit, puisque ce sont essentiellement des **valeurs** qui nous intéressent, les structures de données seront utilisables en tant que propriétés des objets, c'est-à-dire en tant que types d'objets. Toutes les informations relatives à la description des structures (fonctions d'accès en particulier) devront être considérées de la même façon, c'est-à-dire en tant qu'élément (ou valeur) d'un objet composé.

D'autre part, on a vu que certaines représentations des structures de données sont pratiquement imposées par la nécessité d'assigner un emplacement connu à un objet : ici, seules les valeurs des objets ayant une signification ou, plus précisément, toutes leurs composantes devant s'exprimer sous forme de valeurs, les objets peuvent être rangés "n'importe où".

En ce qui concerne la représentation des structures de données dans une mémoire associative, il faut éviter la tentation d'une démarche ascendante : en effet, constatant la bonne adéquation des structures de données aux mémoires adressables, on pourrait penser que les mémoires associatives doivent susciter des structures de données spécifiquement adaptées.

En fait, si l'on impose de respecter pour le support associatif, comme pour le support adressable, une indépendance entre le niveau fonctionnel et la représentation matérielle, cela implique :

- les structures courantes sont utilisables **sans modification fonctionnelle** ; le changement de technique de stockage ne doit pas avoir d'incidence à ce niveau.
- on peut permettre à l'utilisateur la définition de structures fonctionnelles conformes à ses besoins : cependant cela restera du ressort du logiciel. Le matériel, transparent à cet égard, devra permettre cette possibilité. Rien n'interdit alors, **compte tenu de ces restrictions,** que les mécanismes associatifs donnent lieu à une structuration différente des données.

En outre, on a remarqué, plus haut, qu'il semblait exister des structures de données **purement utilitaires** dont le seul intérêt était d'optimiser l'ordonnancement des objets dans l'espace physique (on les avait qualifiées d'"artificielles"). L'absence d'adressage élimine cette

contrainte et l'on peut donc espérer réduire, voire supprimer, la nécessité de leur utilisation.

Dans une mémoire adressable, on traitait de façon différente les valeurs proprement dites et les aspects logiques de la structure (fonctions et paramètres d'accès). Ici, puisqu'on ne traduit plus ces derniers en termes d'emplacements physiques, on devra les considérer comme des valeurs "banalisées". Il faut donc envisager que tous les objets élémentaires de la mémoire aient une forme standardisée, quelle que soit leur signification pour le logiciel (valeur ou fonction d'accès) :

- les valeurs, types d'objets structurés et paramètres d'accès sont tous traités sur le même plan par les mécanismes associatifs, ce sont des propriétés des objets.
- puisqu'on n'utilise pas la topographie de l'espace mémoire, les objets sont rangés "n'importe où".
- les fonctions associées aux structures sont du ressort du logiciel et doivent utiliser des mécanismes associatifs "standardisés".

Exemple : Si deux objets de la mémoire ont une partie de leur valeur égale à "DUPONT" :

- dans une mémoire adressable, c'est leur emplacement qui les distinguait (par l'intermédiaire de leurs fonctions d'accès) et permettait de savoir que l'un était en fait une feuille d'arbre binaire, et l'autre le 356è élément d'une liste.
- Ici, on aurait 2 objets dont la valeur contiendrait :

"FEUILLE D'ARBRE BINAIRE, DE VALEUR : DUPONT"
et : "356è ELEMENT DE LISTE, VALEUR : DUPONT"

Remarques : On préciserait, en outre, le nom des objets ARBRE et LISTE dont font partie ces éléments, la localisation du premier dans l'arbre, etc ...

. la représentation réelle serait sans doute différente de cette simple description fonctionnelle.

Donc, du point de vue du traitement associatif, **toute valeur rencontrée sera soumise aux mêmes mécanismes logiques.** C'est le logiciel, grâce à la description fonctionnelle des structures de données, qui prévoiera que, dans tel objet structuré, telle valeur a une signification particulière

concernant la description logique de la structure.

Ainsi, par exemple, on pourrait imaginer (parmi d'autres possibilités) qu'une matrice plane soit représentée par des triplets (numéro ligne, numéro colonne, valeur) dont les trois composantes sont considérées au niveau matériel comme des valeurs. De même, on pourrait exprimer un chaînage en créant des couples de valeurs des éléments deux à deux consécutifs (ce qui conduirait alors à une répétition de chaque valeur).

Les **avantages** qu'on peut attendre de la **banalisation des valeurs des objets**, et des mécanismes qui opèrent sur elles sont les suivantes :

- aucune règle topographique n'est nécessaire.
- transparence au niveau matériel des traitements associatifs, puisqu'on ramène tout traitement à la manipulation standardisée des valeurs .
- stockage "en vrac" des informations, ce qui implique :
  . interchangeabilité des emplacements vis-à-vis du contenu.
  . à condition d'être suffisante, toute place disponible est utilisable, indépendamment de sa situation.
  . moindre coût du stockage, dans la mesure où le support peut être exploité plus efficacement.
- l'utilisateur (directement ou par l'intermédiaire du système d'exploitation) n'est pas soumis à des contraintes liées aux fonctions d'accès, dans la mesure où celles-ci sont implicitement contenues dans les valeurs.

On peut noter quelques **inconvénients** :

- nécessité d'une représentation explicite des informations logiques structurelles (paramètres d'accès, ...), d'où une certaine perte de place.
- abandon de certaines facilités qu'offraient les relations entre adressage et représentation des structures de données.

Deux observations tendent à minimiser ces inconvénients (outre les avantages déjà cités):

- l'expression de relations de type associatif entre les objets est indépendante de leur représentation, ce qui réduit l'intérêt du stockage explicite d'informations structurantes. Autrement dit, la spécificité du

traitement associatif permet de supposer que l'utilisation d'objets structurés soit plus fortement liée à un réel besoin de telles structures, en tant que **types d'objets** particuliers. Dans ce cas, les relations topologiques entre objets sont de moindre importance et n'ont pas besoin d'**être** exprimées physiquement si un mécanisme de traitement associatif peut permettre de les déduire. A la limite, on peut considérer que le traitement associatif permet, entre autres, de faire apparaître des relations entre les objets sans qu'elles aient été prévues a priori.

- d'autre part, la représentation en mémoire adressable des structures de données pouvait déboucher également sur des informations superflues d'un point de vue strictement fonctionnel (tables, pointeurs, chaînons, ...), dans le cas où certaines facilités de manipulations sont souhaitables. Ces informations, nécessaires à la gestion des emplacements, sont inutiles dans une mémoire associative, ce qui compense en partie la nécessité de représenter explicitement les fonctions d'accès sous forme de valeurs.

En outre, s'agissant d'un système associatif particulier, les exigences que l'on a énumérées peuvent être assouplies. Il serait vraisemblable que les types de données puissent être liés, au niveau élémentaire, à la représentation matérielle, dans un souci d'accroissement de l'efficacité, sans pour autant faire interférer les niveaux fonctionnels et matériels. Ainsi, on peut imaginer que toute structure linéaire de données puisse être prise en charge de façon privilégiée par une implémentation matérielle en mémoires séquentielles.

Du point de vue fonctionnel, cependant, on conservera l'idée d'un espace d'objets standardisés au niveau des valeurs qui les composent d'où toute référence topographique est absente, et dont seules les manipulations externes concerneront l'utilisateur.

II-2 - DÉFINITION ENSEMBLISTE D'UNE MÉMOIRE ASSOCIATIVE

D'après ce qui précède, on peut faire une analogie entre la collection d'objets "en vrac" que constitue la mémoire, et les définitions de la théorie des ensembles.

Précisons cependant qu'il ne s'agit pas ici de définir un modèle mathématique rigoureux, ce qui dépasserait le cadre de cette étude. Cette approche permet simplement de suggérer une image plus formelle et d'inspirer des mécanismes de fonctionnement. En outre, bien que parfois il conviendra de se méfier des imprécisions dues au fait qu'**une analogie n'est pas un modèle**, on pourra emprunter à cette théorie des ensembles son vocabulaire.

La mémoire, ou plutôt, l'espace des informations est un ensemble d'objets dont les types (donc une partie de la sémantique) sont définis par l'utilisateur.

Cet espace des informations, puisqu'on a vu que le traitement associatif excluait en grande partie l'idée de topographie, peut donc être considéré comme une **"boîte noire" où les objets sont "en vrac"**.

Pour traiter comme on le souhaite toutes les composantes de l'objet (valeurs, type, paramètres) sur le même plan, on peut recourir à la notion de **sous-ensemble**.

Les propriétés des objets liées à la notion de "type" ou de "structure" sont alors des **fonctions caractéristiques d'appartenance** à un sous-ensemble. Autrement dit, l'énoncé d'une (ou plusieurs) propriété(s) revient à définir un sous-ensemble des objets de la mémoire.

Exemple : Soit T un arbre binaire de chaînes de caractères. Considérant la représentation de l'objet T dans une mémoire adressable, il s'agissait à la fois d'un objet T, et de la structure fonctionnelle permettant de manipuler des chaînes de caractères utilisant le contexte topographique de la mémoire physique. En d'autres termes, T était à la fois objet et procédure.

Ici on le considèrera comme un objet T appartenant au sous-ensemble dont la fonction caractéristique s'énonce "être du type arbre binaire de caractères".

Cependant, il faut pouvoir discriminer les éléments constitutifs d'un objet structuré et les manipuler selon les fonctions attachées à cette structure. Ces objets élémentaires seront traités de façon homogène, par des définitions du même ordre.

Pour caractériser un élément d'un objet composé, il faut en effet ; - sa valeur intrinsèque

- l'objet composé auquel il appartient
- les paramètres (indices, rang, ...) situant cet élément dans la structure.

Ces caractéristiques seront considérées toutes comme des propriétés associatives ce qui présente l'avantage d'une définition homogène des objets.

Rappelons que dans le cas des mémoires adressables, paramètres et type se traduisaient par les emplacements. Ici, toutes les informations seront utilisées comme des valeurs, les spécifications fonctionnelles étant du ressort du logiciel.

exemple ; Reprenons notre arbre T. Nous considèrerons ses noeuds comme des éléments du "sous-ensemble des noeuds de l'arbre T". La fonction d'appartenance est simplement "être un noeud de l'arbre T". Les éléments de ce sous-ensemble sont des objets comprenant :

- le nom (considéré comme une valeur) T
- la valeur intrinsèque (ici on avait dit qu'il s'agissait de chaînes de caractères)
- des paramètres (rang, droite, gauche) qui seront considérés comme des valeurs par le traitement associatif.

Remarque : Si nous reprenons cet exemple, l'objet composé "arbre T" semble avoir une existence double dans la mémoire :

- un objet du sous-ensemble des arbres binaires
- un sous-ensemble des **noeuds** qui le composent.

C'est-à-dire que tout objet composé peut être considéré comme un objet appartenant au sous-ensemble des objets de même type, et comme un sous-ensemble d'objets élémentaires.

Si nous abordons ce problème sous un autre angle on peut dire que la mémoire est un référentiel composé d'objets élémentaires (indivisibles).

Ces objets sont caractérisés par des propriétés associatives qui permettent de définir des sous-ensembles.

Les objets composés sont en fait des sous-ensembles munis d'une structure. Contrairement aux objets élémentaires, qui font partie de M (le référentiel), les objets composés font partie de $P(M)$, ensemble des parties de M. L'aspect ambigu des objets composés dépendra de la façon dont ils seront utilisés : ils peuvent être intéressants globalement, en tant qu'objets ; ou au contraire, ils peuvent être considérés plus simplement comme le sous-ensemble des objets qu'ils contiennent, si l'on désire accéder à l'un d'entre-eux. On a donc en quelque sorte une manipulation à 2 niveaux : d'une part, on manipule les objets éléments de M, d'autre part les sous-ensembles, éléments de $P(M)$.

Remarquons qu'inversement, il sera commode d'assimiler le sous-ensemble $\{X\}$ dont les propriétés caractéristiques ne sont vérifiées que par le seul objet X, et cet objet X lui-même.

Cette mémoire contenant un "ensemble" d'objets dont on ignore les emplacements physiques peut être considérée ainsi qu'on l'a dit, comme une boîte noire. Il s'agit maintenant d'imaginer de quelle manière l'utilisateur peut manipuler cet ensemble d'objets.

II-3 - MANIPULATION D'UN ENSEMBLE DE DONNÉES ASSOCIATIF

II-3.1. Problème du référentiel

Dans tout traitement ensembliste, il convient de préciser dans quel référentiel on se situe.

En informatique, l'espace des informations vues par un processeur est finalement limité d'une part, par la **taille de la mémoire physique**, d'autre part,le **contexte sémantique** du problème. En effet, dans un problème de traitement numérique, une chaîne de caractères n'a aucune signification. Disons qu'elle ne fait pas partie de l'espace manipulable par le problème.

En outre, dans ce cas, si le référentiel "théorique" est l'ensemble $\mathbb{R}$ des réels, il est bien connu qu'un calculateur ne contient qu'un sous-ensemble fini de $\mathbb{R}$ (taille de la mémoire). De plus, les objets manipulés, en raison de la précision limitée de la représentation choisie, ne couvrent qu'une partie finie des éléments de $\mathbb{R}$.

Dans un autre domaine, si l'on considère l'ensemble des mots que l'on peut former avec les lettres de l'alphabet, il est de taille infinie. Cependant un grand nombre des combinaisons obtenues sont sans intérêt sémantique. Dans le cadre d'une utilisation pour le traitement de texte, on pourra considérer que le référentiel est le vocabulaire français. Dans le cas d'une gestion de fichiers du personnel, pour des raisons d'économie de traitement on peut éliminer des mots tels que CARBURATEUR, ETYMOLOGIE ou SATURNE que l'on est sûr de ne pas devoir envisager.

Finalement, le référentiel est soumis à deux contraintes situées aux deux extrémités de la chaîne de traitement :

- d'un côté, l'utilisateur par le contexte sémantique des problèmes envisagés définit l'espace théorique des informations
- de l'autre, le matériel impose une restriction de cette définition (c'est-à-dire un sous-ensemble du précédent) sous l'influence de deux critères : le nombre d'objets (taille de la mémoire)
  la précision de leur représentation.

Examinons à présent, toujours en référence à notre analogie ensembliste, quelles fonctions ou opérations sont envisageables.

Il est à noter que, puisqu'à un objet on associe un certain nombre de propriétés servant de fonctions caractéristiques, il existe une dualité entre l'aspect global de l'ensemble d'informations et le calcul booléen. Cependant on a vu qu'il serait souhaitable qu'une mémoire associative possède des possibilités plus riches que les mécanismes logiques.

II-3.2. Appartenance

Une fois précisé le référentiel, l'énoncé d'une propriété quelconque p_i pour un objet X est équivalente à l'appartenance de cet objet au sous-ensemble noté F_i dont la fonction caractéristique est p_i. C'est-à-dire :

$$X \in F_i \Leftrightarrow p_i(X) = \text{vrai}$$

Notons qu'il s'agit là de l'appartenance à un sous-ensemble de l'espace théorique des données.

En particulier, si l'on considère les objets effectivement contenus en mémoire comme un sous-ensemble M du référentiel E, l'énoncé de la propriété "être présent en mémoire" permet de vérifier la présence ou l'absence de l'objet dans la mémoire.

II-3.3. Opérations entre sous-ensembles

Toutes les opérations classiques sur les sous-ensembles sont équivalentes aux opérations booléennes sur les propriétés.

Citons notamment :

Intersection

$$X \in F_i \cap F_j \Leftrightarrow P_i(X) \text{ A } P_j(X) = \text{vrai}$$

On voit donc que l'énoncé de plusieurs propriétés définit des sous-ensembles de cardinalité moins importante. Du point de vue associatif, cela permet notamment une sélection plus ou moins stricte des objets.

Union

$$X \in F_i \cup F_j \Leftrightarrow P_i(X) \vee P_j(X) = \text{vrai}$$

A l'inverse, cette propriété permet une certaine latitude dans les critères de sélection, ce qui permet d'élargir le champ d'investigation en cas d'échec.

On pourrait également parler du complément (négation de la propriété correspondante), de la différence, de la différence symétrique.

En fait, toutes ces opérations correspondent aux **mécanismes booléens** rencontrés habituellement dans les mémoires associatives. Nous noterons simplement qu'il y a assimilation entre le sous-ensemble (contenant un seul objet) dont la fonction caractéristique est l'intersection logique de toutes les propriétés vérifiées par cet objet et l'objet lui-même.

Les fonctions d'accès consistent donc à vérifier la présence d'un objet dans la mémoire ou,par l'énoncé de propriétés, de rechercher des éléments appartenant au même sous-ensemble qu'un comparande donné.

On utilise donc ici les traitements associatifs habituels de comparaison d'égalité avec ou sans masquage.

II-3.4. Dénombrement - Cardinalité

Même lorsque l'espace théorique des informations potentielles est infini, la mémoire contient un nombre fini d'objets.

En particulier, pour une (ou plusieurs) propriété(s) donnée(s), il est possible de considérer le nombre d'objets que compte un sous-ensemble.

D'autre part, il est possible (pour des raisons de fiabilité) de vouloir limiter le nombre de réponses à une opération.

La notion de nombre d'objets interviendra donc de deux façons :

1) En tant que **réponse à une demande**, c'est-à-dire que le nombre d'objets est l'information recherchée par l'utilisateur : la question serait du type "Quel est le nombre d'objets vérifiant p et q ?" ce qui équivaut à :

$$\text{Card}\,[F_p \cap F_q]$$

2) En tant que **paramètre d'une demande** c'est-à-dire que l'utilisateur limite le nombre d'objets que contient la réponse : la question pourrait être : "Fournir 3 objets vérifiant p", ce qui équivaut à :

$$\exists\ ?\ A,\ A \subset F_p \ \underline{\text{et}}\ \text{Card}[A] = 3$$

Notons que cet aspect implique un problème de choix des objets fournis. Ce problème est classique en traitement associatif. Il est connu sous le nom de "multiple-match resolution" et plusieurs algorithmes sont envisageables (voir I-2.3.).

II-3.5. Fonctions évoluées

Jusqu'ici, dans l'analogie ensembliste utilisée, hormis l'intérêt d'une relative formalisation, on n'a fait que présenter les mécanismes associatifs les plus courants.

On a parlé plus haut des possibilités "intelligentes" que pourrait offrir une mémoire associative. Examinons de quelle façon il est possible de les intégrer.

II-3.5.1. *Prise en charge de la ressemblance. Propriétés métriques*

Ici on va s'intéresser aux objets individuellement.

En général, il est commode d'évaluer une ressemblance entre 2 objets par l'intermédiaire de la notion de distance entre les valeurs possédées par les objets. En effet, il est plus simple de quantifier la dissemblance que la ressemblance.

La ressemblance est donc définie par

$X\ R\ Y$ ssi $d(X, Y) \leq \lambda$ λ est le degré de ressemblance

où $d : E \times E \rightarrow \mathbb{R}$ (éventuellement on peut se restreindre, de $\mathbb{R}$ à $\mathbb{N}$)

notons que la relation de ressemblance (contrairement à la similitude, cas où $\lambda = 0$) n'est pas une relation d'équivalence; en effet, elle n'est pas transitive :

Exemple : si $\mathcal{R}$ est définie par "deux mots français qui n'ont pas plus d'une lettre différente"

alors RAMEUR $\mathcal{R}$ RUMEUR

RUMEUR $\mathcal{R}$ HUMEUR

mais on n'a pas RAMEUR $\mathcal{R}$ HUMEUR.

La prise en charge de la ressemblance suppose donc la disponibilité d'un mécanisme de mesure_de_distance.

La distance de deux objets étant fortement liée à la sémantique du problème, c'est l'utilisateur qui définira le calcul des distances entre objets.

Cependant dans l'hypothèse (vraisemblable) où l'on disposera d'objets de base (types de base), les distances entre ces objets seront fixées de façon standardisée. De même que les objets sont constructibles à partir de types de base, les distances servant à comparer ces objets seront bâties à partir des distances élémentaires.

La distance entre 2 objets, (de même que le cardinal d'un sous-ensemble) pourra intervenir, soit en **réponse** à une demande, soit en **paramètre**. Les questions faisant intervenir la distance pourront donc être de deux sortes :

1) "Quelle est la distance entre X et Y ?"

2) "Existe-t-il un (ou plusieurs) objet(s) X_i tel(s) que $d(X_i, C) \leq \lambda$? "

II-3.5.2. *Voisinage. Propriétés topologiques*

Bien que, par principe, le contexte associatif s'interdise le recours à la topographie des emplacements de mémoire (adressage), il faut pouvoir prendre en compte les aspects topologiques inhérents à l'espace d'informations manipulé.

Ainsi, la notion de distance qu'on vient d'introduire implique-t-elle des propriétés topologiques dans l'ensemble des données. Notons toutefois que la topologie décrite ici peut différer quelque peu d'une topologie rigoureuse au sens mathématique.

Rappelons encore, en effet, que le raisonnement tenu ici vise principalement à appliquer la **terminologie** de la théorie des ensembles à la description de la mémoire associative. Les nombreuses correspondances qu'on a pu établir ouvrent sans doute la voie à une modélisation mathématique précise qui dépasserait le cadre de cette étude. C'est pourquoi les définitions introduites, par souci de simplification, peuvent perdre, dans une certaine mesure, la rigueur des définitions mathématiques.

1) Si l'on fait intervenir la notion de distance en tant que paramètre d'une question, on pourra appeler **voisinage d'un objet X** le sous-ensemble de E défini par :

$$V_\lambda(X) = \{\lambda \in E \ / \ d(X, Y) \leq \lambda\}$$

A cette occasion, on peut revenir sur la notion de distance : pour une distance d, il faudra évidemment que Y, pour faire partie du voisinage de X, soit comparable à X, c'est-à-dire que d(X, Y) ait un sens. De ce fait, toute distance induit dans E deux parties disjointes :

- les objets pour lesquels d a un sens
- ceux qui ne peuvent être évalués à l'aide de d.

2) Si dans un sous-ensemble donné, il est possible de définir un **élément "nul"**, la distance de tout objet à cet objet de référence permet de calculer une **norme**. Par exemple, en numérique, la valeur absolue d'un entier est une norme.
Dans le contexte du traitement de texte, si l'on prend la chaîne "AA ... A" comme référence, l'application pondérée (de gauche à droite) qui mesure l'écart lettre à lettre d'un mot quelconque à cette référence peut être considérée comme une norme, utilisée pour déterminer le rang alphabétique.

3) La disponibilité d'une norme induit immédiatement l'idée d'une **relation d'ordre**. En reprenant les exemples précédents : on sait que,

pour la valeur absolue, -5 est "plus grand que" +2,et "plus petit que" -7 ou +7.
De même, pour la norme "rang alphabétique", on a :

ASSOCIATIF < TRAITEMENT
AUTOMATIQUE < INFORMATIQUE

4) De la relation d'ordre, on débouche sur la notion d'**élément maximal ou minimal** d'un sous-ensemble. On dira :

$MIN(F) = x \in F, \forall y \neq x \in F, y > x$
$MAX(F) = x \in F, \forall y \neq x \in F, y < x$

Tout cela ne représente qu'une énumération hypothétique des possibilités que l'on peut imaginer a priori.

On s'est jusqu'ici intéressé à la façon dont l'utilisateur pouvait définir son interprétation fonctionnelle des données mémorisées et de quels moyens il pourrait disposer pour les manipuler. En somme, nous avons considéré l'ensemble des données du point de vue des **actions** qu'il était possible d'exercer sur lui.

On peut, en contrepartie, adopter le point de vue des **résultats** que peut fournir la mémoire, c'est-à-dire, en fait, les informations que l'on peut en attendre.

II-4 - INFORMATIONS FOURNIES PAR LA MÉMOIRE

II-4.1. Réponses intrinsèques

La classe d'informations les plus couramment restituées par une mémoire est évidemment les objets traités eux-mêmes. C'est pourquoi on a adopté le qualificatif d'"intrinsèques" : toute réponse à une manipulation de ce type sera constituée d'un objet ou d'un sous-ensemble d'objets.

Les applications réalisées utilisent surtout ce mode de fonctionnement ; c'est notamment le cas de la restitution associative de données, ou des traitements numériques de tableaux (voir chapitre I).

Remarquons que l'introduction de paramètres nouveaux (cf. utilisateur) peut permettre d'enrichir l'aspect intelligent de la mémoire. Autrement dit les paramètres fournis pour obtenir des réponses intrinsèques concourent à l'amélioration des possibilités de choix permettant de sélectionner ces réponses.

Dans ce qui a précédé, les opérations purement booléennes sont évidemment de ce type. Cependant, on peut considérer que l'appartenance (vérification de la présence d'un comparande en mémoire) est extrinsèque puisqu'elle produit un état booléen.

II-4.2. Réponses extrinsèques

Au contraire des précédentes, les informations demandées ne sont pas des objets propres de l'espace des données.

Remarquons que, du point de vue du traitement, ces opérations seront plus riches puisqu'elles doivent élaborer des informations externes à partir d'objets de l'espace mémoire ; il ne s'agit donc plus ici à proprement parler de mémorisation, mais bien de possibilités de traitement intégrées à la ressource mémoire. Cela implique notamment de décrire ces traitements à la mémoire associative sous une forme à définir ; câblage, microprogrammation, procédures ? ... En tout cas, il semble certain qu'une partie de cette description doive être accessible

à l'utilisateur.

Dans ces informations extrinsèques on pourrait ranger :

1) Les évènements (booléens)

- présence/absence d'un objet en mémoire
- conjonction /disjonction de sous-ensembles :

par exemple : "est-ce que $F \cap G = \emptyset$"

- contenance :

par exemple : "est-ce-que $F \subset G$?"

(notons qu'il s'agit d'un ordre sur $\mathcal{P}(E)$)

- similitude :

"existe-t-il Y, tel que $X \equiv Y$"

- relations d'ordre :

Ce sont des comparaisons entre objets, par référence à une norme (voir II-3.5.2.)

"est-ce-que $X < Y$"

etc ...

N.B. Remarquons au plan de la réalisation que la transmission de la réponse sera très facile puisqu'un bit suffira. On peut peut-être imaginer de spécialiser une ligne pour ce genre d'évènements.

2) Les mesures

Ce seront le plus souvent des nombres entiers (parfois des rationnels). Citons par exemple :

- Cardinal d'un sous-ensemble
- distance entre 2 objets
- norme d'un objet
- éventuellement masse (ou poids) d'un objet.
- taux de ressemblance ; ce seront par exemple les scores probabilistes utilisés en reconnaissance automatique de la parole (cf. chapitre IV).

II-4.3. Remarque

Les possibilités citées ne sont nullement exhaustives : pour en permettre d'autres (liées à l'optimisation d'une application) le matériel devra permettre au logiciel d'"inventer" les modes de fonctionnement.

Remarquons par exemple que la notion de masse permet d'envisager des opérations telles que le calcul du centre de gravité d'un sous-ensemble d'objets : cela peut permettre de définir la "moyenne" d'un sous-ensemble ce qui peut présenter un intérêt dans certaines applications. En reconnaissance automatique de la parole, notamment, l'aspect compression de données est important : il permet par exemple de réduire la dimension de l'espace traité ou de baser une méthode de reconnaissance sur la détection de traits pertinents (analyse en composantes principales, formants [107], energoïdes [108]), ...).

II-4.4. Notion de rendement

Finalement, la mémoire associative se comportera comme un système auquel l'utilisateur fournit des éléments sous forme de questions. Le système fournit en retour d'autres informations qui constituent les réponses. On néglige ici (sans l'oublier) l'étape d'initialisation (transitoire) que constitue le chargement et pendant lequel le transfert d'informations ne se fait que vers la mémoire.

La richesse de ces échanges d'informations (du point de vue de l'utilisateur) est en quelque sorte le rendement du système.

Sans aller jusqu'à aborder la théorie de l'information, on peut considérer que le rapport entre la complexité de la question et la richesse de la réponse peut permettre d'évaluer intuitivement le rendement.

La notion de pertinence de la question semble être du ressort de la théorie des questionnaires (cf I-543).

A priori, il semble que la qualité sémantique de la réponse soit inversement proportionnelle au nombre d'objets qu'elle contient, ou au "poids sémantique" (à définir) de l'évènement quelle annonce.

D'autre part, la question est d'autant plus complexe que le nombre de paramètres spécifiés est grand.

Sous toutes réserves, on pourrait considérer que le rendement R s'exprime par le ratio :

$$R = \frac{\text{"Quantité" d'information de la réponse}}{\text{"Quantité" d'information fournie dans la question}}$$

La notion de "quantité" est évidemment liée à la sémantique des données.

Le "rendement" de la mémoire associative devra donc subir deux influences :

- pertinence du logiciel (cf. théorie des questionnaires)
- "intelligence" du matériel.

Le premier point débouche sur l'étude d'un langage adapté à l'utilisation efficace de cette mémoire, le critère étant l'optimisation de l'interrogation. Notons d'ailleurs que c'est aussi la succession des interrogations qu'il convient d'optimiser en tenant compte des réponses obtenues.

Le second point implique de prévoir des mécanismes évolués au sein de la mémoire ce qui enrichit les informations stockées. L'architecture adoptée devra effectuer un compromis entre :

- richesse des traitements, intégrés à la mémoire, disponibles
- souplesse et universalité d'emploi.

En effet, toute tendance d'intégration de traitement au câblé implique une plus grande rigidité de l'emploi.

II-5- TYPES DE MANIPULATIONS

De ce qui précède, en se référant aux informations obtenues par l'utilisateur, nous pouvons distinguer trois types de fonctions.

II-5.1. Requêtes

Elles correspondent à ce que nous avions appelé réponses "intrinsèques"

les **informations fournies sont des objets** de l'espace manipulé.

les **paramètres** utilisés sont :

- des sous-ensembles définis par un jeu de propriétés : cela correspondra de façon élémentaire à présenter un comparande partiellement masqué.

- éventuellement des informations "extrinsèques", les mesures : ce seront par exemples des grandeurs telles que : distance, norme, cardinal, ...

Les opérations ensemblistes ($\cup$, $\cap$, complément, ...) seront possibles de deux façons :

. soit en utilisant la séquentialité des requêtes par la possibilité d'accumuler des résultats.

. soit en utilisant le parallélisme des requêtes par la possibilité de combiner logiquement les paramètres.

Ces deux manières de synchroniser les opérations débouchent inévitablement sur l'opposition rapidité-complexité matérielle.

Exemple : union de 2 sous-ensembles.

Il s'agit de définir F $\cup$ G dont les propriétés caractéristiques sont p(F) et p(G). On aura donc 2 solutions :

- une question unique sur le critère "p(F) $\vee$ p(G)"
- les deux questions successives "p(F)" puis "p(G)"

Les requêtes correspondent finalement aux accès d'une mémoire adressable mais l'utilisation de propriétés associatives leur confère une puissance supérieure.

Outre les opérations booléennes entre sous-ensembles citons comme requêtes :

- voisinage (objet) au sens de la distance précisée
- minimum / maximum au sens d'une relation donnée.

Contrairement aux opérations prises en charge par le matériel ce genre de requêtes sera en fait probablement une véritable séquence (procédure, µ-programme ? ...)

II-5.2. Calculs

Les informations fournies sont "extrinsèques". On les appellera des **mesures** (ce seront des **nombres**).

Leurs paramètres sont des **objets** (en fait, ce sont, classiquement des opérandes).

La description du calcul dépendra de la complexité du matériel. De plus, l'organe qui assumera cette fonction prendra en charge une partie seulement des opérations.

Ainsi il est probable que le calcul du cardinal d'un sous-ensemble soit assumé par le matériel (par exemple, il serait possible de compter les marques d'activation).

Par contre, certains calculs, en raison soit de leur complexité soit du degré de liberté laissé à l'utilisateur, devront être décrits par programme (éventuellement µ-programme).

Ici encore, il conviendra d'évaluer le compromis entre l'efficacité et la complexité du matériel.

Remarquons que les requêtes peuvent prendre en compte des paramètres "extrinsèques" (mesures) alors que les calculs ne portent que sur des objets.

II-5.3. Prédicats

Les informations fournies sont également "extrinsèques". On les appellera des **évènements** (ce seront des **booléens**).

Leurs paramètres sont des **objets** et, éventuellement, des mesures.

Typiquement, il s'agira de comparaisons mettant en jeu soit des relations d'équivalence (similitude) soit des relations d'ordre (>, <, ...)

Outre ces comparaisons, on peut envisager des évènements comme "sous-ensemble vide" ou des anomalies comme "requête illicite", "mesure impossible", "objet dupliqué", "dépassement de capacité", etc ... Ces véritables indicateurs d'état du système dépendront en fait des choix logiques et matériels que l'on fera par la suite.

L'avantage de ces prédicats est de fournir une information puissante à l'aide d'un seul état booléen. Cependant, leur nombre étant limité, il conviendra de les choisir judicieusement notamment ceux que l'on voudra implémenter de façon câblée. Ce problème est à rapprocher du mot d'état d'un processeur classique.

II-5.4. Remarque

Pour les 3 types de fonctions décrits, il faudra faire intervenir des critères économiques (notamment coût/performances) pour situer la frontière entre les possibilités prises en charge par le matériel et les fonctions qui devront donner lieu à une décomposition parallèle ou séquentielle.

On a donc amalgamé jusqu'ici des fonctions qui seront implémentées sous forme d'opérations câblées, des instructions, c'est-à-dire des combinaisons parallèles de ces opérations, et des macro-instructions, c'est-à-dire des séquences d'instructions.

Autrement dit, les primitives pourront apparaître sous forme câblée, sous forme de micro-programme, etc ...

II-6 - DÉCOUPAGE FONCTIONNEL DU SYSTÈME

Sans préjuger des choix que l'on sera conduit à effectuer ultérieurement aux niveaux logique et matériel, il est possible de dégager dès à présent une sorte d'épure de l'architecture du système.

En effet, la description faite des fonctions prises en charge par le système entraîne à priori des implications sur la structure fonctionnelle du système.

Nous nous sommes essentiellement placé du point de vue des demandes d'informations de l'utilisateur. Nous avons alors distingué trois types de fonctions : il semble commode de leur faire correspondre 3 blocs fonctionnels du système :

- les requêtes sont prises en charge dans la partie stockage du système. L'**unité de stockage** est assez classiquement composée d'un réseau de mémoire associative comportant une logique simple distribuée.
- les calculs, qui fournissent les informations que nous avons appelées mesures, sont effectués par une **unité arithmétique**. Celle-ci devra sans doute être spécialisée, en vue du type de calculs requis (dénombrement, distance, etc ...)
- les prédicats, qui renseignent sur l'état du système ou fournissent des informations booléennes, seront évalués par **l'unité de contrôle**. Ce ne sera sans doute pas la seule fonction de cette dernière.

Sur la figure 14, on a représenté ces trois blocs, avec les fonctions assurées et les divers types d'informations susceptibles d'entrer en jeu. Nous n'avons évidemment mentionné que les flux de données. Il conviendrait d'y ajouter notamment les commandes : ainsi par exemple, l'interprétation et/ou décodage des messages donne lieu, par l'intermédiaire de l'unité de contrôle, aux signaux nécessaires à la spécification des opérations effectuées par l'unité arithmétique.

Deux autres blocs ont leur importance : d'une part, il est nécessaire d'interpréter les demandes reçues sous forme de messages de la part du système-hôte. D'autre part, comme dans tout système associatif, on est confronté au problème des réponses multiples (voir I-2,3,). Cela impose

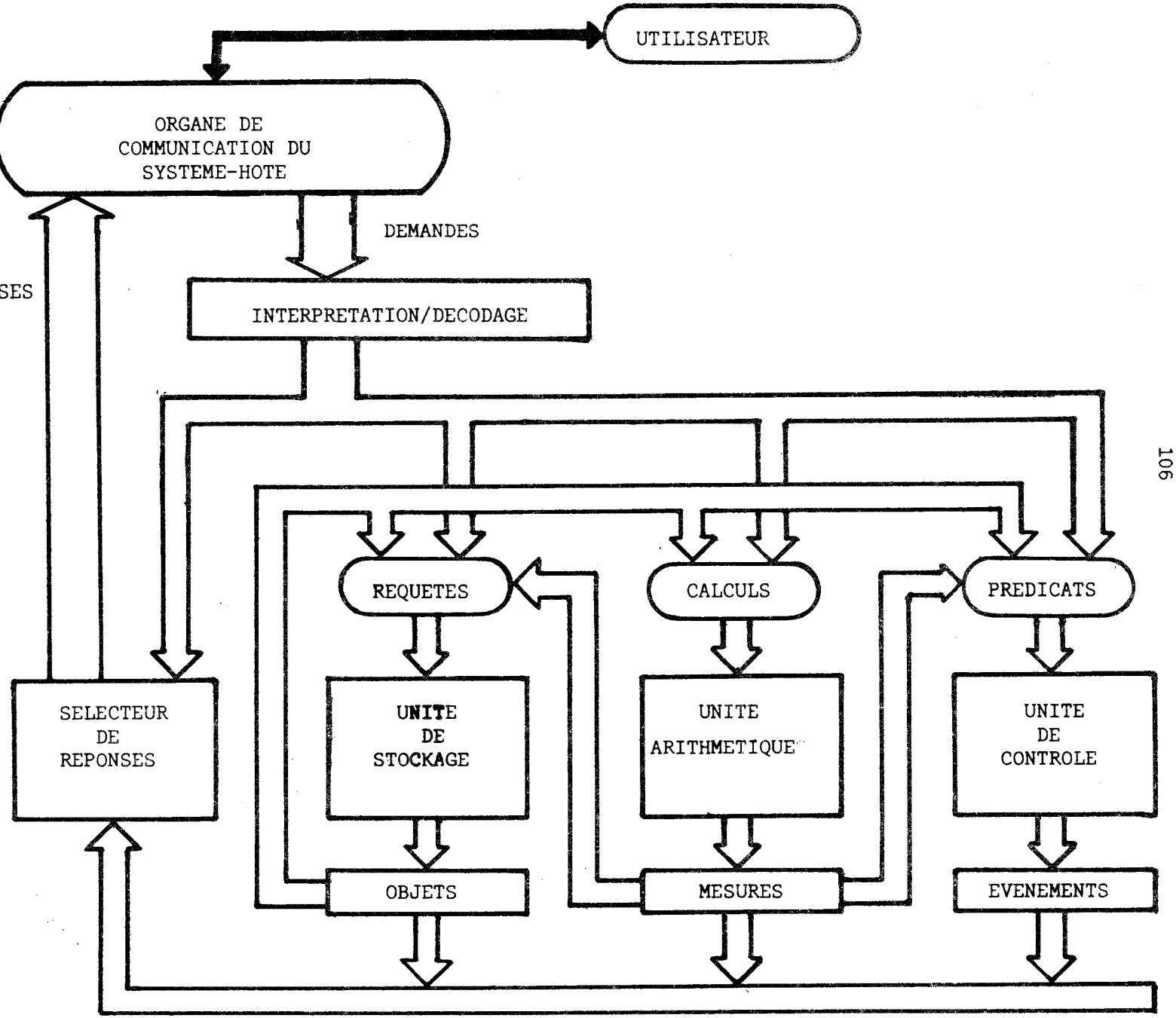


Fig. 14 : Découpage fonctionnel

de prévoir un organe permettant de sélectionner une ou des réponse(s). La complexité de cet organe dépendra naturellement de l'algorithme adopté au niveau des réponses multiples. Une des solutions consiste à effectuer un OU logique sur toutes les réponses, ce qui est très simple à condition que le résultat obtenu corresponde à ce que l'on désire. D'autres solutions simples, mais peut-être contraignantes et appauvrissantes, consistent à choisir arbitrairement, par inhibition matérielle, une réponse unique.

Nous avons déjà précisé que l'on n'a effectué qu'un découpage fonctionnel du point de vue des services rendus à l'utilisateur. Il conviendrait de lui ajouter, outre les commandes que l'on a évoquées plus haut, les facultés de chargement, de programmation (ou plus vraisemblablement de micro-programmation) d'échange avec le système de communication, etc .. Signalons qu'un des problèmes courants de ce genre de systèmes est celui de l'initialisation et de l'allocation de l'espace mémoire libre.

Dans la suite, il est probable que les divers blocs que nous avons défini ici ne seront pas aussi clairement distincts. Nous serons sans doute amenés à faire certains regroupements ou certaines séparations au niveau matériel. Cependant, la description faite ici devrait rester valide tant au plan des fonctions qu'au plan de la circulation des divers flux d'informations.

CHAPITRE III

ORGANISATION LOGIQUE
D'UN SYSTEME ASSOCIATIF

Dans cette partie, on s'appuie sur la description fonctionnelle abstraite pour étudier quel genre d'organisation logique pourrait traduire les concepts énoncés dans cette description.

On envisage tout d'abord des suggestions liminaires dont il serait possible de tirer parti : il s'agit plus d'options éventuelles que de principes catégoriques. Un système particulier ne saurait d'ailleurs les prendre toutes en compte, mais on retrouvera plus loin certains prolongements des notions ainsi évoquées.

On pose ensuite des hypothèses sur la représentation des objets. Celle-ci implique un assez large éventail de requêtes possibles, que l'on essaie de caractériser.

On peut alors discerner deux familles de méthodes d'interrogation associative. Cette distinction relève aussi bien du type de tâche envisagé que des solutions matérielles possibles.

On examine alors de quelle façon une architecture multiprocesseurs pourrait prendre en charge les cas où une interrogation exhaustive de la mémoire s'avère nécessaire et ce, avec un temps de réponse incompatible avec un matériel classique.

III-1 - REMARQUES PRÉLIMINAIRES

La description fonctionnelle qui précède s'est voulue délibérément abstraite, de façon à conserver un certain recul par rapport aux contingences relatives à l'implémentation.

Cependant, il convient dès à présent d'exprimer un certain nombre de considérations qui tendent à moduler cette description. Compte-tenu de la multitude des options possibles (cf. chapitre I) dans la conception d'un système, il s'agit plus de suggestions faites "a priori" que de caractéristiques que l'on respectera impérativement.

III-1.1. Aspects liés à la sémantique de l'utilisation

III-1.1.1. *Notion de "type"*

Dans de nombreux cas, les tâches manipulent à un premier niveau des **types d'objets**. En général, à un type de données est associée une collection d'**opérations spécifiques**.

On pourrait donc imaginer de traiter à un premier niveau, privilégié, la propriété définissant le type. Cette propriété de "type" serait prise en charge par une sorte d'**antémémoire associative**. Un mécanisme câblé (du style pagination) associerait au type reconnu le "bloc" mémoire (ou la page) concerné.

On réaliserait pratiquement une **partition typologique** du système, ce qui pourrait présenter plusieurs avantages :

1) Accélération de l'accès grâce à l'antémémoire : en effet, le type dans un ensemble d'objets hétéroclites, est une propriété déterminante, et donc on réduit notablement ainsi la quantité de cellules à examiner.

2) Protection des objets vis-à-vis de certaines opérations : on peut en effet rendre impossibles certaines opérations entre objets de blocs (et donc de types) différents. De même on peut assurer la validité d'une opération autorisée localement.

3) Accélération du traitement : on peut en effet envisager d'effectuer simultanément des opérations dans des blocs distincts du système (M.I.M.D.) dans la mesure où les unités de traitement, associées à des types d'objets différents pourraient être indépendantes.

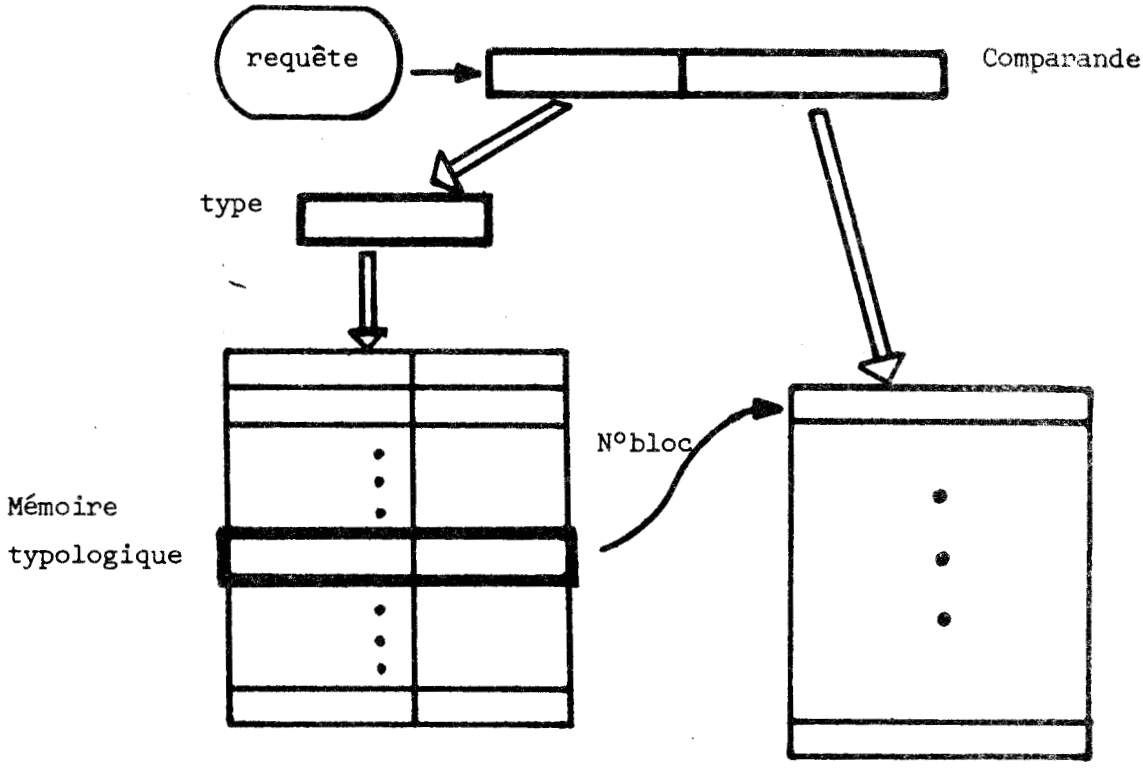
Notons que le partitionnement du système peut être plus ou moins poussé, c'est-à-dire qu'il peut intervenir à un niveau plus ou moins grossier des tâches prises en charge. Donnons deux possibilités d'intervention de cette notion:

- assez simplement, l'utilisation du "type" peut être considérée comme un **argument d'accès**. Son implémentation est alors analogue à un système paginé mais peut être améliorée en fonction des informations manipulées. Le but essentiel est alors de réduire le temps de recherche de l'information (figure 15 a)

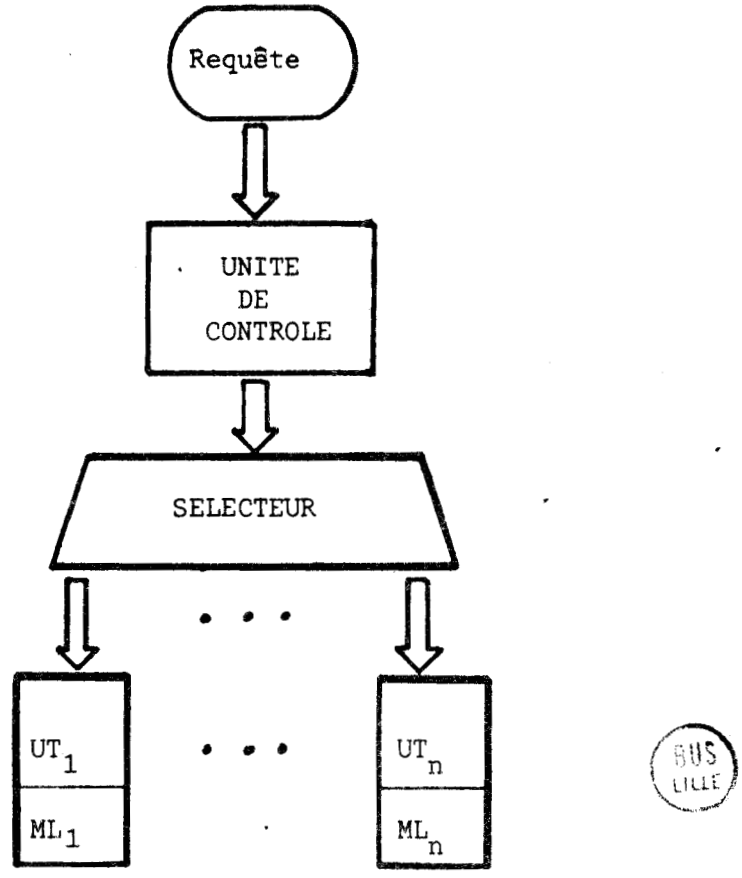
- plus globalement, on peut considérer un système **multiprocesseur** dans lequel chaque unité de traitement est chargée d'un type spécifique de données, c'est-à-dire en définitive d'une famille d'opérations. Une unité de contrôle interprète les demandes et les route vers les unités de traitement correspondantes (figure 15b).

Cet usage de la notion de type est évidemment lié au genre d'utilisation envisagé et doit tenir compte des caractéristiques générales du système. Cependant nombre d'applications, a priori, pourraient recourir à cette faculté.

Ainsi, par exemple, citons le cas de l'identification des segments phonétiques dans le cadre de la reconnaissance automatique de la parole (voir chapitre IV). En effet, beaucoup de systèmes recourent à une classification grossière — ne serait-ce qu'une discrimination "voisé/non voisé"— pour orienter la suite du traitement. On peut donc penser, par exemple, que les types "voyelle" ou "consonne" donnent accès à deux parties distinctes de la mémoire, ou mieux, dans un système multiprocesseurs, fassent intervenir deux sous-systèmes séparés. Remarquons, toujours sur cet exemple, que la typologie, selon les cas, peut alors être raffinée : ainsi, dans les consonnes, distingue-t-on plusieurs classes, fricatives, plosives, sourdes ...



a. Dans un monoprocesseur (par "pagination")



b. Dans un multiprocesseur

Fig. 15 : Utilisation de la notion de type

Un inconvénient dans ce genre d'approche est que la propriété intitulée "type" doit être suffisamment déterministe et sûre pour que le choix effectué au Premier niveau puisse être fiable. D'où une difficulté pour déterminer cette propriété, c'est-à-dire pour établir une hiérarchie définitive sur l'ensemble des données manipulées. Cela impose une connaissance précise a priori de cet ensemble, et un seul mode d'accès, toujours selon la même hiérarchie des critères.

III-1.1.2. *Succession des requêtes*

Nous nous intéressons ici à des accès "aléatoires" successifs aux objets mémorisés.

En général, compte-tenu de la sémantique de chaque application il existe entre 2 accès mémoire successifs une **corrélation** non négligeable.

Dans les mémoires adressables, on utilise cette propriété (connue sous le nom de principe de localité) d'autant que l'aspect séquentiel des algorithmes de traitement permet presque une anticipation statique des accès nécessaires. Finalement, cela relève de la méthode d'implantation en mémoire des programmes et des données.

Dans le cas de requêtes associatives, au niveau des valeurs mêmes on peut faire une observation analogue :

Si à l'instant t_i, la requête X_i a donné la réponse $R(X_i)$, la réponse $R(X_{i+1})$ à la requête X_{i+1} (instant t_{i+1}) est contenue dans un sous-ensemble de la mémoire qui peut être prédéfini.

Si on peut associer à $R(X_i)$ un sous-ensemble F_i dans lequel on soit sûr que $R(X_{i+1})$ figure, on peut ainsi s'affranchir de l'investigation de $C(F_i)$.

Le fait que deux réponses se suivent est induit par le déterminisme inhérent aux requêtes. Ce déterminisme est introduit soit par un phénomène physique (acquisition et traitement de signaux) soit par l'utilisateur (interrogation de bases de données).

Dans le second cas, il est possible de prévoir a priori ce genre d'évènement, et certaines constructions de bases de données peuvent

en tenir compte.

Par contre, dans le cas où le phénomène générant les informations est quasi-aléatoire ou, ce qui revient pratiquement au même, est trop complexe pour être pris en compte, l'observation de cette propriété se fera surtout a posteriori. Toutefois il est très souvent possible de répertorier un certain nombre de "cas de corrélation" qui peuvent permettre d'améliorer les performances.

On peut alors imaginer plusieurs façons d'utiliser cette caractéristique ; nous donnerons deux exemples possibles :

- mémorisation d'une (ou plusieurs) requête(s) précédente(s). Les ambiguités ou les incertitudes entachant la réponse $R(X_{i+1})$ peuvent éventuellement être réduites par la connaissance de $R(X_i)$ (voire $R(X_{i-1})$... etc ...). Il s'agit là d'un procédé de type "retour arrière"

- prise en charge des "successeurs" (sémantiques) d'un objet. La réponse $R(X_i)$ fournit ainsi le sous-ensemble $S(R(X_i))$ dans lequel on doit trouver $R(X_{i+1})$. Dans ce cas, on travaille inversement au cas précédent par anticipation ou prédiction.

En pratique, ces deux types de méthodes doivent pouvoir représenter pour chaque objet A_i le sous-ensemble $S(A_i)$ des "successeurs possibles" de A_i. Cela peut impliquer des techniques de chaînage, linéaires ou multi-chaînons, ou encore l'usage de matrices de succession (figure 16). Ainsi dans le cadre de la reconnaissance automatique de la parole, le système MAP (SILVERMAN [109], par exemple, utilisait cette notion en explorant une matrice de transition représentant les successions possibles entre phonèmes (appelées "transèmes").

$R(X_i)$ \ $R(X_{i+1})$	A_o	A_1	. . .	A_k	. . .	A_N
A_o	S_{oo}	S_{o1}		S_{ok}		S_{oN}
A_1	S_{10}	S_{11}		S_{1k}		S_{1N}
.			.		.	
.			. . .		. . .	
.			.		.	
A_m	S_{mo}	S_{m1}		S_{mk}		S_{mN}
.			.		.	
.			.		.	
.			. . .		. . .	
.			.		.	
A_N	S_{No}	S_{N1}		S_{Nk}		S_{NN}

Fig. 16 : Matrice de succession

- Les S_{mk} peuvent être simplement binaires (succession possible ou non), ou peuvent représenter des probabilités (ce qui imposera l'usage d'un seuil).

- Notons que la matrice peut être monolithique, comme ici, ou au contraire éclatée dans la mémoire, chaque ligne (S_{mo}, S_{m1}, ... S_{mN}) pouvant être associée à chaque A_m. Dans ce cas, on se rapproche alors des techniques basées sur le chaînage.

En tous cas, fonctionnellement, la succession des requêtes se traduit par un chaînage des accès, que ce soit directement ou par l'intermédiaire d'une matrice. Le processus des accès successifs peut alors se schématiser de la façon suivante (figure 17) :

si on suppose que $A_m \in R(X_i)$

alors $R(X_{i+1}) \subset S(A_m)$

où $S(A_m)$ est le sous-ensemble des successeurs de A_m.

Insistons encore sur le fait qu'un tel procédé suppose une pré-connaissance approfondie des données prises en charge.

En outre, la réduction du temps de réponse que ces techniques permettent d'espérer doit être justifiée : elle se fait au prix d'un accroissement de la complexité du stockage, voire d'une redondance des informations mémorisées (c'est également le cas de certaines techniques de bases de données) En effet, dans le mesure où l'on doit recourir, dans une telle hypothèse, à l'adressage, la seule utilisation de liens de chaînage est pénalisante.

Dans certains cas, la taille mémoire n'est pas une contrainte - soit que le nombre et la longueur des données soient faibles, soit que l'on puisse (en fonction du temps de réponse) utiliser des mémoires de masse -. On peut alors imaginer une partition de la mémoire en sous-espaces d'après la succession des réponses possibles. Chaque sous-espace correspond à l'un des arguments A_m et contient le sous-ensemble $S(A_m)$ des "successeurs possibles" de A_m, éventuellement assortis d'une probabilité de succession. Cette solution, évidemment coûteuse en espace mémoire, représente la stricte implémentation du schéma de la figure 17.

On travaille alors à 2 niveaux. Au niveau de prédiction, on associe à la requête (X_{i+1}) le (ou les) sous-espace(x) à explorer. Au niveau association, dans ce (ou ces) sous-espace(s) on extrait le (ou les) $R(X_{i+1})$ correspondant(s).

Notons alors qu'il serait plus efficace d'utiliser dans cette hypothèse une structure multi-processeurs. Chaque processeur élémentaire (rappelons ici l'hypothèse selon laquelle le nombre d'objets est suffisamment faible) se verrait confier un sous-espace pour la phase d'association. Un processeur-maître assumerait le niveau prédiction ; il gérerait la table de succession pour mettre au travail tel ou tel processeur, recueillerait les réponses $R(X_i)$ fournies par ces processeurs, et éventuellement opèrerait une sélection sur ces réponses (nombre, seuil, ...). En outre, dans ce cas, on n'alourdirait pas le travail par le recours à l'adressage ; en effet, chaque PE (processeur élémentaire) est purement associatif et donc ne nécessite pas de mécanisme d'adressage interne.

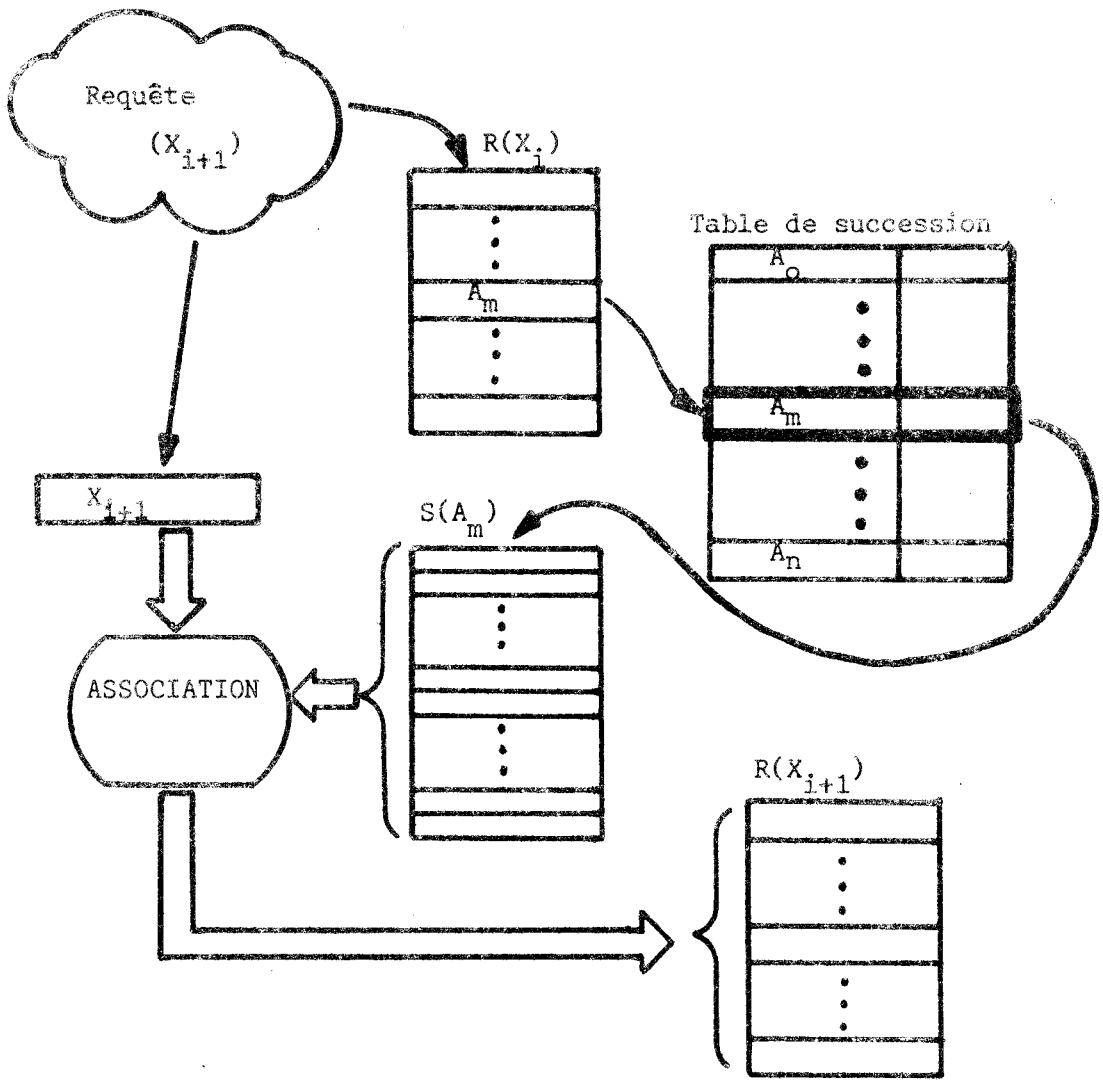


Fig. 17 : Utilisation des successions de requêtes

Remarque : Dans le cas le plus simple, la méthode d'interrogation est telle que $R(X_i)$ ne contient qu'un seul des $A_m (0 \leq m \leq N)$. Sinon, il faut opérer un tel processus pour chaque $A_m \in R(X_i)$.

La seule sélection est faite au niveau du processeur-maître pour associer l'un des PE_m à un objet A_m contenu dans un sous-ensemble de réponse. Toutefois, là encore, la sophistication d'une telle structure matérielle demanderait à être justifiée par l'application envisagée.

Remarquons enfin que les deux notions précédentes, "type" et "succession des requêtes" s'expriment toutes deux par le même genre de solution, partition de la mémoire d'un monoprocesseur (c'est-à-dire utilisation de l'adressage) ou éclatement sur un multiprocesseur (c'est-à-dire introduction d'un certain parallélisme). C'est dire que la même structure matérielle pourrait assumer indifféremment les deux aspects. Or, les structures induites par l'une ou l'autre de ces approches seront le plus souvent distinctes : autrement dit, dans la plupart des cas, et ce en fonction du type de données traitées, on utilisera l'une de ces propriétés à l'exclusion de l'autre. Cette souplesse est alors grandement facilitée par leurs implications identiques au niveau de la structure matérielle.

III-1.2. Aspects liés à l'implémentation

III-1.2.1. *Caractéristiques et performances du système*

Nous avons maintenant établi (cf. Chapitre I) qu'un système associatif ne pouvait être envisagé "pro arte", mais, au contraire, devait s'insérer dans une solution plus globale à un problème bien défini.

De ce fait, les performances obtenues devront être proportionnées aux impératifs liés à la tâche à effectuer. En conséquence, une solution matérielle originale ne se justifiera que par un **accroissement "utile" des performances** par rapport à une implémentation logicielle sur un monoprocesseur classique.

Dans le même ordre d'idées, dans la description fonctionnelle (voir chapitre précédent), ont été introduites des notions telles que "objets","requêtes", "prédicats", "calculs", ... Celles-ci se verront assumées à un niveau plus proche soit du matériel, soit du logiciel,selon d'une part la **rapidité** nécessaire d'autre part la **souplesse** souhaitée. Ainsi, il est problable que des "calculs" élémentaires soient disponibles

de façon câblée, les autres devant être programmés ou microprogrammés.

D'autre part, deux types de contraintes vont également intervenir au niveau des caractéristiques générales du système :

1) Vis-à-vis de l'"extérieur", c'est-à-dire de son environnement : en effet, le système associatif est envisagé comme une machine spécifique dans un système plus complet. Dans un monoprocesseur, ce serait une ressource spécialisée voire pratiquement un périphérique "évolué" ; dans un multiprocesseur, il s'agirait d'un des processeurs spécialisés, au même titre que, par exemple, un gérant de base de données ou un processeur flottant. Outre une connexion relativement aisée au niveau matériel, il faut pouvoir assurer une utilisation suffisamment simple de ce sous-système par le reste du système. C'est-à-dire que, au niveau du logiciel, celui-ci puisse mettre en oeuvre le processeur associatif par un jeu de commandes simples et puissantes, de façon analogue aux commandes adressées à un contrôleur de périphérique. Mieux, on peut comparer le système associatif à un canal : dans ce cas, celui-ci interprète les quelques instructions réservées de la machine-hôte qui le concernent et exécute le programme-canal correspondant. Un avantage supplémentaire est de rendre l'utilisation par la machine-hôte du système associatif indépendante des possibilités câblées de celui-ci.

2) Vis-à-vis de l'"intérieur", c'est-à-dire de la technologie mise en oeuvre : on a déjà mis en évidence la corrélation entre les différentes technologies et les performances. Corollairement, la structure du système est liée au type de solution matérielle adoptée : ainsi, un moyen de stockage séquentiel impliquerait nécessairement un certain degré de sérialisation dans l'interrogation de la mémoire. Inversement, un système associatif totalement parallèle nécessiterait forcément une technologie de stockage permettant au moins l'accès aléatoire, voire une technologie tout à fait originale , imbriquant cellules de stockage et portes logiques et/ou arithmétiques.

III-1.2.2. *Influence de la consécutivité physique*

Nous avons admis que l'accès associatif aux objets se traduirait ← du moins en excluant l'hypothèse de la simulation d'une mémoire associative →

par l'abandon, lors de l'accès, de toute référence aux positions physiques des informations.

Cependant, en envisageant une implémentation, il semble intéressant de tenir compte de la simple succession des informations, c'est-à-dire leur **position relative**. Cela ne remet pas en cause le principe rappelé ci-dessus, dans la mesure où la position absolue des objets restera inutilisée.

Autrement dit, les objets ne sont plus vraiment "en vrac" mais sont, au moins partiellement, organisés linéairement en séquences. Cela présente quelques avantages :

1) D'abord, si l'on considère l'**implémentation** : il n'existe pas, actuellement, pour des raisons évidentes de conception et de fabrication, de support de stockage "en vrac". Dans toutes les techniques de stockage, les cellules de mémorisation sont organisées linéairement, soit par adresses auxquelles on accède aléatoirement, soit par positions de circulation auxquelles on accède séquentiellement par une fenêtre unique. Donc, par la force des choses, les objets seront stockés sous formes de files ou d'anneaux, d'autant que, pour des raisons de coût, des mémoires circulantes représenteraient le meilleur compromis. Il faudra tirer parti de cette séquentialité imposée au niveau matériel.

2) Les **structures de données** (qui restent bien sûr du ressort du logiciel) représentent une caractérisation importante des objets stockés. Or, la plupart de celles utilisées habituellement sont facilement adaptables à une organisation séquentielle : le plus simple est par exemple de se servir de valeurs réservées pour jouer le rôle de délimiteurs ou d'informations syntaxiques. Les structures linéaires (listes, files, ...) sont évidemment les plus favorisées de ce point de vue. Toutefois, les arbres, par exemple, peuvent être représentés en utilisant uniquement des délimiteurs.

3) Certaines **propriétés sémantiques** de l'utilisation (cf. III-1.1.) pourraient tirer profit de cette organisation. Ainsi, par exemple, pourrait-on traduire les relations de succession sémantique entre objets par une consécutivité physique, ce qui tendrait à améliorer le temps de réponse. Une autre approche pourrait consister à regrouper en séquences physiques

des objets de même "type" : dans ce cas, pour pouvoir tirer profit de cette propriété il faudrait pouvoir accéder par le "type" à la séquence correspondante ce qui tend à ressembler à un accès aléatoire. Cependant, on l'a vu, cela peut correspondre à une partition de l'espace mémoire en blocs distincts à accès sélectif.

Ainsi la caractéristique physique de structure linéaire des cellules de stockage, loin d'être une restriction par rapport à la description fonctionnelle du système associatif, s'avère comme une source d'avantages possibles. Ces avantages seront ressentis tant au niveau matériel (coût du stockage et absence d'adressage) qu'aux niveaux logiciel (prise en charge des structures de données) et pragmatique (prise en charge de propriétés relatives à la sémantique de l'utilisation).

Dès lors, cela débouchera sur deux caractéristiques générales de l'implémentation :

1) Les objets sont organisés séquentiellement —le plus généralement sans imposer de relation d'ordre, mais cette possibilité peut se révéler intéressante à exploiter—. On peut donc aisément traduire la notion de sous-ensemble par des **séquences d'objets** (éventuellement disjointes).

2) La primitive d'accès élémentaire consiste à passer d'un objet à son **successeur** dans la séquence. Cela peut éventuellement correspondre à un mécanisme physique (cas des mémoires circulantes). Ce mécanisme est nécessaire et permet, par exemple, la lecture d'un objet structuré : il suffit alors d'appliquer itérativement cette primitive entre les deux "valeurs" délimitant le début et la fin de cet objet.

III-1.2.3. *Gestion de l'espace physique de stockage*

L'organisation séquentielle, dès à présent pose cependant une difficulté, liée à l'absence d'adressage et donc l'impossibilité d'utiliser des chaînages : il s'agit de la **récupération d'emplacements** en cas de mise à jour ou de **débordement**. Une solution complète à ce problème ne peut se concevoir sans référence précise à la structure matérielle définie pour l'implémentation. On peut cependant faire ici quelques observations à ce sujet :

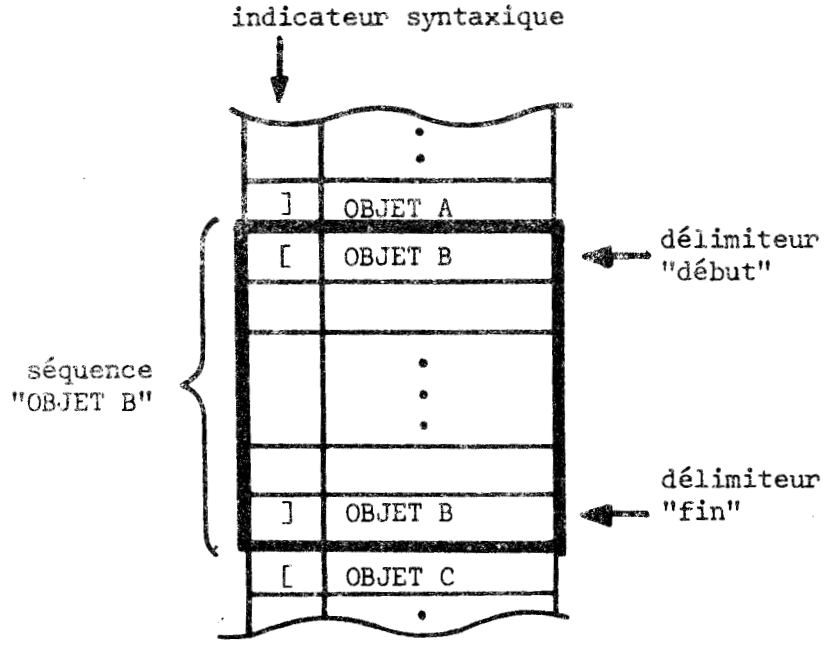
Tout d'abord, ce problème de gestion des emplacements mémoire disponibles, s'il est classique dans le cas des mémoires adressables (encore que souvent esquivé ...), n'a été pratiquement pas abordé explicitement dans le cas des mémoires associatives. En effet, on s'intéresse dans ce cas essentiellement à manipuler des valeurs plutôt que des adresses : de ce fait, le problème se pose en termes différents :

La position absolue des **objets pris individuellement** n'a aucune importance : les ajouts éventuels peuvent donc se faire n'importe où dès lors qu'on trouve un emplacement disponible.

Dans le cas des **objets structurés** (où la position relative joue un rôle) un ajout éventuel peut se concevoir de deux manières : ou bien l'on restructure et on redéplace l'objet structuré dans sa totalité si cela s'avère nécessaire, ou bien, si cela ne remet pas en cause la structure, on tolère que l'objet structuré soit stocké en 2, voire plusieurs morceaux. Cela serait tout à fait imaginable pour des séquences d'objets (telles qu'évoquées au paragraphe précédent) : les délimiteurs de fin de la première sous-séquence et de début de la deuxième sont alors remplacés par des valeurs spéciales jouant le rôle de "lien associatif" (voir figure 18). Notons que cette technique deviendrait plus lourde dans le cas de plus de 2 sous-séquences (ordre des sous-séquences) et qu'en outre elle ralentit le temps de réponse. Il ne s'agit que d'un exemple : la banalisation de toutes les informations (objets élémentaires, délimiteurs, indices, informations syntaxiques, ...) devrait permettre au niveau du logiciel d'adapter une solution appropriée à l'utilisation.

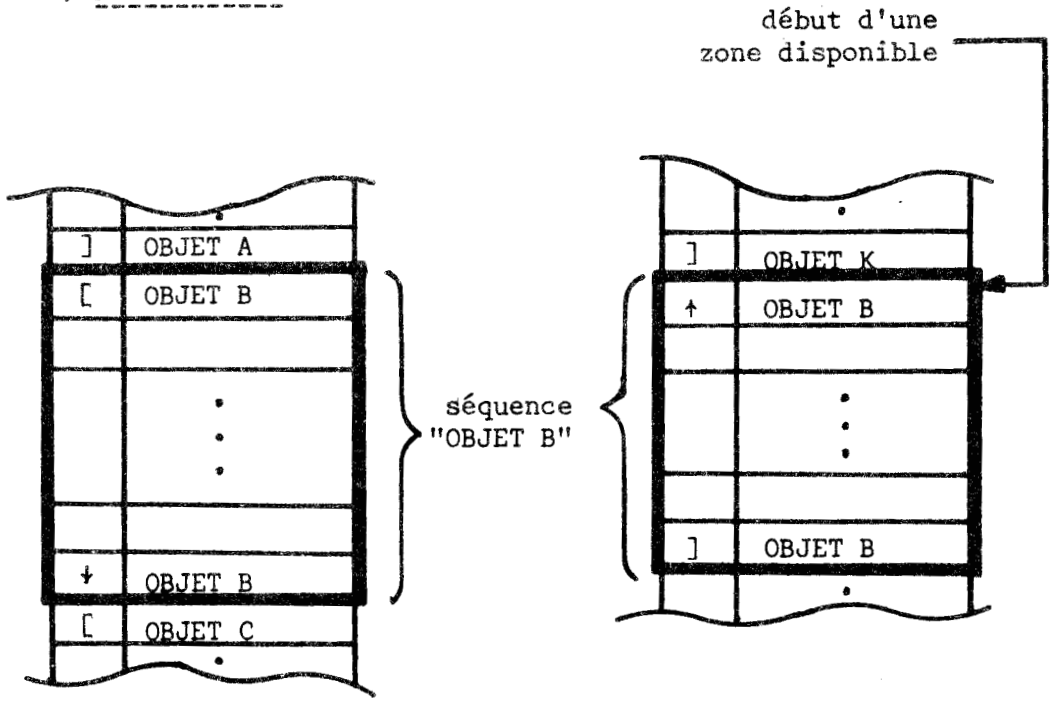
Plus délicat apparaît le problème de la **récupération des emplacement disponibles.**

En effet, dans le cas des mémoires adressables (voir par exemple PAIR & GAUDEL [103]), on pourra recourir à des techniques telles que le **chaînage des places libres**. Sans détailler plus, précisons qu'il faut parvenir à empêcher un émiettement trop important de l'espace mémoire et/ou une croissance excessive des zones de débordement. Les techniques alors mises en jeu relèvent plus généralement du système d'exploitation : quand cela est possible, une **réorganisation** périodique de la mémoire (qu'elle soit principale ou secondaire) s'avèrera bénéfique.



N.B. Une partie de la valeur sert à représenter des indications syntaxiques :
- [,] symbolisent ici les délimiteurs,
- ↓, ↑ symbolisent les "liens associatifs"

a) Etat initial



b) Après allongement de la séquence "OBJET B"

Remarques : - En cas d'insertion de valeurs dans la séquence "OBJET B", un décalage est nécessaire : il faut pouvoir sauvegarder au fur et à mesure les valeurs déplacées.

- Dans le cas d'un éclatement en plus de 2 morceaux, il faut prévoir des liens distincts permettant, au besoin, de restaurer l'ordre des sous-séquences (on peut préférer interdire ce cas et imposer des restructurations).

Fig. 18 : Un exemple de mise à jour (adjonction) par utilisation d'un "lien associatif"

Dans le cas des **mémoires accessibles par le contenu**, au contraire, la plupart de ces techniques sont caduques. On a vu (chapitre I) que certains systèmes réservent un indicateur "disponible" pour signaler les cellules libérées par la suppression d'une information : cependant, celles-ci survenant de façon éparse, on peut craindre d'arriver à un émiettement.

Une réponse complète à ce problème doit prendre en considération les caractéristiques physiques de la mémoire (ainsi que son organisation) et les contraintes (performances notamment) imposées par l'application. Autrement dit, avant toute implémentation particulière, on adoptera des attitudes proportionnées au caractère plus ou moins crucial de cette question. De ce fait, sans préciser davantage (car cela repose sur une implémentation éventuelle), on peut dégager plusieurs voies d'approche :

. Tout d'abord, dans le cas où il est possible d'affirmer que **la taille mémoire est un problème négligeable**, cette question de prise en charge des emplacements libres devient fort peu embarrassante : à la limite, on peut presque l'ignorer. Tout au plus, pourra-t-on prévoir, épisodiquement, une réorganisation de la mémoire. Cette attitude est analogue au cas des bandes magnétiques, où, la plupart du temps, l'accent étant mis sur le débit et la capacité "virtuelle", on se soucie assez peu d'utiliser les blocs éventuellement libérés de façon clairsemée. Eventuellement à des moments où la charge du système le permet, on effectuera une copie de bande pour compacter les informations.

. Une façon quelque peu différente d'éviter ce problème est de s'intéresser au genre de traitement lui-même. En effet, dans de nombreux cas, il est possible de considérer que **les mouvements sont peu fréquents** voire inexistants. Ce sera le cas notamment lorsqu'on manipule des données telles que des dictionnaires, des références en reconnaissance de formes, etc ... En fait, il semble a priori que la plupart des tâches associatives puissent être vues sous cet angle. Dès lors, l'espace mémoire devient physiquement analogue à une mémoire "morte", dans laquelle l'organisation est uniquement conçue pour favoriser les accès en consultation. On rejoint presque l'idée précédente, dans le sens où une mise à jour (éventuelle mais très rare) ne serait alors permise que par le biais d'une ré-initialisation.

. Les cas où ce problème devra être réellement pris en compte par le matériel semblent donc très rares : on peut cependant supposer que cette fonction supplémentaire implique des organes matériels spécifiques : ainsi, un processeur spécifique doté de **tampons** et de **compteurs** serait-il sans doute nécessaire : encore faut-il adapter les caractéristiques matérielles (longueur des tampons notamment) aux tailles des zones que l'on souhaite pouvoir récupérer. De toutes façons, un tel processeur doit être justifié par une nécessité impérieuse et ne peut s'envisager qu'en relation avec la conception des autres parties de l'implémentation. On peut également penser qu'on peut tirer parti, si besoin est, d'une organisation originale de la mémoire permettant par exemple une taille physique variable. Ainsi des idées comme une mémoire hiérarchisée "souple", ou la mémoire logarithmique (voir VANLAER [61]) pourraient-elles être exploitées.

Toutefois, il semble plus simple de considérer le traitement d'une mémoire associative comme relevant des deux premières approches évoquées. En effet, les deux principaux contextes d'application qu'on peut envisager confirment ces hypothèses :

- dans le cas des **bases de données**, les accès sont essentiellement des consultations, et compte-tenu des temps de réponse exigés, on peut se satisfaire d'un éparpillement des données dans le support physique.

- dans le cas des problèmes de **reconnaissance de formes**, pour la partie confrontation aux références, seule une réponse rapide est intéressante. Les modifications éventuelles sont à rejeter au niveau de la phase d'initialisation.

Du point de vue de la taille mémoire, finalement, ces deux applications sont analogues dans la mesure où le temps de réponse reste proportionné au volume des informations consultées ; seule, l'organisation du stockage (structure et procédé physique) permet d'assurer les possibilités de manipulation souhaitées ; ce ce fait, toute *réorganisation* ne peut s'envisager que comme une *réinitialisation du système*.

III - 2 - DÉFINITIONS

III-2.1. Représentation logique des objets

On a vu qu'un objet de la mémoire associative pouvait être défini comme une intersection de **propriétés caractéristiques**. Ces propriétés seront manipulées de façon associative.

En ce qui concerne l'implémentation, il est plus simple de supposer un aspect banalisé à la représentation de ces propriétés de façon à les traiter globalement de façon standardisée. C'est le logiciel qui interprêtera et prendra en charge leur contenu sémantique.

On peut donc représenter une propriété par une variable prenant ses valeurs dans un ensemble de définition dont les éléments sont les différents états possibles de cette propriété.

Il est donc commode de traduire chaque **propriété associative** par une telle variable. On considérera donc qu'un objet est représenté par un vecteur de valeurs. On les appellera les attributs de l'objet.

Représenter un objet consiste donc à assimiler :

objet = $\{P_i / Pi$ sont des propriétés associatives$\}$

à:

objet = vecteur d'attributs

Dans le cas le plus général, la dimension du vecteur n'est pas fixée (on peut alors considérer cette dimension comme un attribut implicite), et les ensembles de définition sont quelconques. On aurait donc des objets qui seraient des points, des droites, des plans, des hyperplans de l'espace E x F x G x ... x Y.

Cependant, dans la plupart des cas on pourra se restreindre (selon l'application ou le type de données) à des espaces tels que $\mathbb{R}^n$ ou $\mathbb{Z}^n$, ou encore $\{0, 1\}^n$.

Dans le cas le plus général, un objet est représenté par un **k-uple** d'attributs :

$$A = (a_1, a_2, \ldots a_j \ldots a_k)$$

soit un vecteur de $E_1 \times E_2 \times \ldots \times E_j \times \ldots E_k$. E_j est fini et son cardinal v_j, est supérieur à 2.

Le choix d'un espace de référence reste évidemment du ressort du logiciel. Cependant, les caractéristiques propres au matériel auront leur influence, notamment en ce qui concerne le compromis entre la **largeur du mot-mémoire** et le **nombre de mots** par séquence physique (ce sont les "dimensions" de la mémoire). En effet, il apparaît souhaitable qu'existe une certaine compatibilité entre les deux niveaux (matériel et logiciel), c'est-à-dire, ici, entre les deux formats que sont la largeur du mot-mémoire et la taille de la représentation d'un attribut.

A la limite, en travaillant sur $\{0, 1\}^n$, avec n aussi grand que nécessaire, chaque attribut devient alors la représentation d'une propriété booléenne, avec : attribut $(p_i) = 1$ (resp. 0) **ssi** p_i = VRAI (resp. FAUX). Autrement dit, tout mot binaire de n bits pourrait être interprété ainsi.

Cependant, dans la majorité des cas, cette décomposition est trop fine, et l'on travaillera au moins sur des octets, ce qui permet de représenter 256 valuations (ou encore 256 lettres d'un alphabet de symboles). La distinction, par rapport au cas précédent, est uniquement conventionnelle : en effet, pour la même longueur en bits, les propriétés seront, ici, beaucoup plus significatives (256 valuations au lieu de 2), mais beaucoup moins nombreuses (8 fois moins).

En fait, le format de base de la mémoire sera une contrainte inévitable. De plus, une certaine standardisation peut simplifier la gestion de la mémoire. On fait donc alors les hypothèses supplémentaires suivantes :

1) Tous les E_j sont égaux à E ; l'espace de référence est alors de la forme E_k.

2) Les v_j sont toutes égales à v(= Card (E)) : le nombre de valuations par attribut est fixe.

On peut remarquer au passage, si on admet que k puisse varier (longueur variable des objets) que la représentation d'un objet est un mot sur l'alphabet E (au sens de la théorie des langages).

L'ensemble E sera déterminé par rapport au format matériel. On peut cependant supposer intéressant de choisir $E = \{0, 1\}^L$, où L est le nombre de bits par mot-mémoire (ou un multiple).

Quel que soit le codage adopté pour la représentation, les nouvelles hypothèses ne constituent pas une restriction trop sévère : ce sera le rôle du logiciel, lors du passage objet ←représentation (resp. représentation ←objet) d'interprêter (resp. de coder) en attributs les valeurs de E (resp. les attributs en valeurs de E).

Dans ce qui suit, tant que cela ne sera pas nécessaire, nous ne ferons plus de distinction entre objet et représentation d'une part, et attribut et valeur d'autre part.

Remarquons, enfin, que ce principe de représentation des objets, par un k-uple **ordonné** de valeurs, serait tout à fait compatible avec l'idée de consécutivité physique exprimée plus haut : un objet serait alors stocké comme une séquence d'attributs.

III-2.2. Calculs et prédicats

Dans la description fonctionnelle, trois types de manipulations de la mémoire associative ont été définis ; Rappelons les succinctement :

- les **requêtes** consistent à retrouver des sous-ensembles d'objets de la mémoire à partir de l'énoncé de propriétés caractéristiques. Les **réponses** fournies sont **intrinsèques** dans la mesure où ce sont des **objets** explicitement stockés dans la mémoire.

- Les **calculs** et les **prédicats**, au contraire, fournissent des **réponses extrinsèques** qui sont des informations implicitement contenues dans la mémoire mais qu'il est nécessaire d'évaluer. On les a appelées respectivement des **mesures** (produits des calculs) et des **événements** (produits des prédicats).

Les **prédicats** peuvent alors être de deux natures :

- d'une part, un certain nombre d'"états" du système ont pu être prévus à la conception. Les évènements fournis constituent alors en quelque sorte le vecteur d'état du système et sont accessibles par l'utilisateur (exemples : dépassement de capacité, erreur, résultat d'une comparaison, ...)

- d'autre part, du côté de l'utilisation, on peut avoir besoin d'évaluer certaines conditions dont l'état n'est pas élaboré par le matériel ; il faut alors les tester par programme (exemple : existence d'un objet, apparition de telle propriété entre 2 objets ou sous-ensembles ..)

Autrement dit, ces interrogations, dont les réponses sont booléennes peuvent être :

- soit prévues par le système (au plan de la conception), auquel cas, en raison de leur caractère général, elles seront disponibles sous forme câblée (ou microprogrammée).

- soit liées à la sémantique (au plan de l'utilisation), auquel cas, il sera nécessaire de les programmer.

De même, en ce qui concerne les **calculs**, leur puissance est un compromis qui tiendra compte :

- de l'organisation du stockage et de la représentation des données, qui influencent largement le rapport coût/performances des possibilités de traitement ; en effet, la répartition de celles-ci dans les sites de stockage (degré de parallélisme) et leur caractère plus ou moins spécialisé jouent à cet égard un rôle important.

- du volume des communications entre la machine-hôte et la mémoire associative ; en effet, un des buts essentiels du transfert de possibilités de traitement vers les sites de stockage est de limiter les échanges d'informations à un dialogue "riche" sur le plan sémantique. De ce fait, le système associatif doit être doté de possibilités suffisamment "intelligentes".

Ces deux impératifs ne sont pas incompatibles. La caractéristique essentielle du système associatif sera de pouvoir effectuer globalement et "simultanément" des traitements sur un grand nombre d'objets : il est alors préférable de disposer de possibilités élémentaires très réparties (ce qui va dans le sens du parallélisme) que d'un processeur évolué mais unique. Les opérations plus complexes auraient alors le désavantage de devoir être programmées. Cependant, elles concernent en général un nombre d'objets plus restreint et il sera donc plus efficace de les réserver au système-hôte qui supportera le logiciel.

Donc, en ce qui concerne l'évaluation des mesures associatives on répartira le traitement de la façon suivante :

. opérations élémentaires/nombre élevé d'objets, dans les sites de stockage (ce qui en outre favorise une adaptation du système à des usages différents).

. opérations complexes/nombre relativement faible d'objets, prises en charge par le système-hôte.

De ce fait, le coût de l'implémentation pourra rester dans des limites raisonnables tout en offrant une réduction sensible des communications entre les deux systèmes et donc un accroissement de l'efficacité.

Avant d'aborder une implémentation —et rappelons que chaque machine associative est un cas particulier—, il n'est guère possible de poser des hypothèses supplémentaires sur ces manipulations de la mémoire associative.

En effet, les calculs et les prédicats, on l'a vu, relèvent, d'une part, des contraintes liées à l'utilisation (type d'informations, type de tâches, ...), d'autre part, des possibilités câblées offertes par l'implémentation.

En conséquence, seul le contexte particulier d'un cahier des charges précis permettra de définir correctement ces fonctions et notamment de situer la frontière entre le câblé et le programmé. On peut toutefois émettre l'hypothèse d'une relation entre la notion de type d'objet et les fonctions associées à ce type ; il est probable en effet

que le matériel prenne en charge les objets à un niveau élémentaire, en particulier, des attributs de nature assez générale comme l'entier, le caractère ou la chaîne de bits. Les opérateurs associés à ces "types élémentaires" seront, de préférence, câblés. La construction d'objets structurés plus complexes et/ou la traduction en des attributs de nature différente relève ensuite du logiciel : il est assez naturel que les opérations associées soient liées à ces constructions et donc décrites par programmes.

Exemple : On suppose qu'une machine prenne en charge des séquences d'attributs et que les seuls attributs connus par le matériel sont des mots binaires (chaînes de bits) ou des entiers sans signe. On suppose alors également que des unités de traitement sont réparties dans le système : ces unités disposent d'opérateurs câblés dont un soustracteur, pour les entiers. Pour construire un objet plus complexe, il faut décrire cet objet par logiciel et éventuellement décrire aussi des fonctions globales sur cet objet. Ainsi on pourra représenter un vecteur d'entiers comme une séquence d'attributs. La "soustraction" de 2 vecteurs doit être définie par programme à partir de la soustraction de 2 entiers.
(Autre exemple, la distance euclidienne :

$$d(V_1, V_2) = \left(\sum_{j=1}^{k} |a_j^1 - a_j^2|^2 \right)^{1/2}$$

$$\text{si } V_i = (a_1^i \; a_2^i \; \ldots \; a_k^i) \qquad)$$

De même, on peut construire des arbres ou des listes de caractères à partir d'un type élémentaire "caractère", et des opérations globales sur ces objets à partir des opérations dont on disposerait pour les caractères.

Les choix des types d'attributs élémentaires et des opérateurs nécessaires sont évidemment liés à l'application et au coût de leur implémentation. Ils ne seront donc envisageables qu'à ce niveau seulement.

III-2.3. Les requêtes

Les requêtes, elles aussi, ont été évoquées d'un point de vue strictement fonctionnel. Comme pour les calculs et les prédicats, toutes

ces fonctions seront, selon leur degré de complexité, prises en charge à différents niveaux de la hiérarchie qui s'étend du matériel au logiciel dans le système.

Ainsi, ces fonctions seront en fait effectuées à l'un des niveaux suivants :

- **opérations câblées**, qui correspondent à prévoir un circuit câblé réalisant en un temps élémentaire une fonction donnée.

- des combinaisons horizontables de ces opérations (par micro-programmation horizontale et/ou logique combinatoire) : ce seront les **instructions**.

- des combinaisons verticales des précédentes (par micro-programmation verticale ou logique séquentielle) : il s'agira alors de **macro-instructions**.

- des **routines logicielles** analogues à un programme-canal ou à une procédure d'E/S, etc ...

Les termes employés ici sont évidemment une simple convention permettant d'illustrer cette hiérarchisation des tâches depuis le câblé vers le logiciel utilisateur. Il est non moins évident que la "puissance" de telle ou telle requête (et donc de l'exploitation du système) est lié directement à sa position dans cette hiérarchie. Inversement celle-ci joue un rôle contraire dans le domaine du coût du système et de son caractère plus ou moins spécialisé.

Intuitivement, pour illustrer ces deux notions, disons qu'une requête est d'autant plus efficace qu'elle est proche du matériel mais qu'elle est alors d'autant plus coûteuse et d'autant moins modifiable.

Cependant, une différence apparaît entre les réponses fournies par les calculs et les prédicats d'une part, et les requêtes d'autre part.

En effet ces dernières produisent des **réponses intrinsèques**, c'est-à-dire que, grossièrement, leur rôle consiste à permettre le repérage (puis la lecture éventuelle) des **objets eux-mêmes** et/ou de

sous-ensembles de ces objets. Avec l'hypothèse adoptée sur la représentation des objets comme des **vecteurs d'attributs** (i.e., pour une implémentation spécifique, des séquences de mots), les requêtes sur ces objets sont indépendantes de l'interprétation de ces attributs.

Ainsi, de même que tout objet manipulé par une tâche peut se ramener à un tel vecteur d'attributs, toute requête portant sur la sémantique du contenu de l'objet pourra se traduire en une requête portant sur un (ou des) attribut(s). Au contraire, dans le cas des calculs, par exemple, la définition des opérateurs n'est pas indépendante du "sens" des attributs.

En conséquence, sur la base de la représentation logique adoptée en III-2.1., il est possible d'examiner quelles peuvent être les **types de requêtes disponibles.**

Dans ce qui suit on appelle $R = E^k$, l'espace de référence dans lequel les attributs prennent leurs valeurs.

Q est l'ensemble des requêtes admises par le système.
Une requête est considérée comme une application de $P(R) \rightarrow P(R)$ (parties de R) on la note $Q_i (\in Q)$.
Le sous-ensemble $Q_i(F)$ est appelé **réponse** à la requête Q_i sur le sous-ensemble F.

Remarque : si on appelle M l'ensemble des objets effectivement contenus dans la mémoire, on aura généralement

$M \subset E^k$ avec $Card(M) << Card(E^k)$.

De ce fait, le sous-ensemble F qui sert de valeur à Q_i est, soit confondu avec M (cas d'une requête sur toute la mémoire) soit une partie de M.
On supposera désormais qu'on travaille sur M ou un sous-ensemble de M : $Q_i(F)$ pourra donc être vide.
On notera également $A = (a_1, a_2 \ldots a_j \ldots a_k)$ les objets où les a_j sont les attributs.

III-2.3.1. Requêtes par identité d'attributs

La comparaison de base sera l'égalité entre un, plusieurs ou tous les attributs des objets de M avec ceux présentés dans la requête. On notera C_i le **comparande** c'est-à-dire l'objet de ℜ qui sert de paramètre à la requête.

1) Identité

Cette requête permet de vérifier qu'un objet est présent dans F.
Dans ce cas $Q_i(F)$ ne comportera qu'un élément. On peut également assimiler cette requête à un **prédicat**, dans la mesure où la réponse peut être booléenne.

2) Requête à attribut unique

$$Q_i(F) = \{A \in F,\ a_j = r\}$$

où r est une valeur telle que $r \in E$.

3) Identité partielle

On dira plus précisément, requête à t attributs. Une requête d'identité partielle Q_i correspond à un comparande C_i où k-t attributs ne sont pas spécifiés.

Cela revient à ajouter dans E une valeur "**indifférent**" qu'on représentera ici par le symbole "⊠".

Alors, on peut définir :

$$Q_i(F) = \{A \in F \ /\ \forall\, j\ (1 \leq j \leq k),\ \underline{\text{Si}}\ C_{ij} \neq ⊠\ \underline{\text{alors}}\ a_j = c_{ij}\}$$

autrement dit, pour k des a_j on a $a_j = c_{ij}$

pour les k-t restants : $c_{ij} = ⊠ \Rightarrow$

$$\forall\, A \in M,\ (a_j = c_{ij}) = \text{VRAI}$$

Le comparande C_i peut également être considéré comme **masqué** pour k-t de ses attributs.

Remarque importante : Les attributs masqués sont définis explicitement. Dans le cas où l'on souhaiterait définir les objets pour lesquels t attributs **quelconques** parmi les k sont identiques aux valeurs du comparande, il s'agirait d'un tout autre type de requête. En effet, les requêtes d'identité partielle sont décomposables en requêtes à attribut unique successives, en se restreignant à chaque étape au sous-ensemble fourni par la précédente, ce qui réduit assez fortement l'espace à explorer. Au

contraire, dans le cas d'une requête à t attributs quelconques, cette approche serait impossible.

Exemple : Si M est le dictionnaire des noms communs français, la réponse à une requête sur un sous-ensemble F ⊂ M pourra être :

$Q_i(F)$ = {RUMEUR, FUMIER, FUMEUR, FUMOIR, HUMEUR, HUMOUR, TUMEUR}

Si Q_i s'exprime par (■ U M ■ ■ R)

III-2.3.2. Requêtes étendues

Ici les critères d'évaluation des attributs sont plus généraux que l'égalité (inégalité, préordre, distance, ...)

1) Identité partielle étendue

On admet, outre le fait de "masquer" certains attributs, que certains de ceux qui caractérisent la requête puissent être validés pour **plusieurs valeurs**, ou pour un **intervalle** donné.

On aura donc une requête analogue au cas précédent, mais avec, par exemple,

$$r_{j1} \leq a_{ij} \leq r_{j2}$$

ou : $Q_i(F) = \{A \in F, (1 \leq a_1 \leq 3) \wedge (a_3 = 5) \wedge (a_6 = 10) \wedge (a_9 = (8) \vee (16))\}$

ou encore en reprenant l'exemple de III-2.3.1., 3)

Q'_i = (F U M ■ ■ R) et (a_4 = I ou ∅)

d'où $Q'_i(F)$ = {FUMIER, FUMOIR}

2) Ressemblance avec seuil

Cette requête impose la définition d'une **distance** sur M, ou, au moins, sur F. Cette distance entre objets s'exprimera le plus souvent grâce à une distance sur les attributs.

La requête de ressemblance avec seuil se compose de deux paramètres :

un objet C_i appelé **comparande** ($C_i \in E^k$)

une valeur λ_i appelée **seuil**

et d'une distance inter-objets notée d(A, A'), évaluée itérativement d'après $d(a_j, a'_j)$ entre attributs.

On aura alors, par définition,

$$Q_i(F) = \{A \in F \ / \ d(A, C_i) \leq \lambda_i\}$$

La valeur λ_i pourra être un réel, mais il sera souvent plus simple de se restreindre à un rationnel, voire à un entier. La distance dépend évidemment du type d'objets considéré. Donnons quelques exemples :

. E = {0, 1} d représente le nombre d'attributs (bits) tel que $a_i \neq a'_i$ (distance de HAMMING)

. E = $\mathbb{R}$ d peut alors être une distance numérique telle que $d(A, A') = \left[\sum_{i=1}^{k} |a_i - a'_i|^2\right]^{1/2}$ (distance euclidienne)

. E = {alphabet de caractères} d peut servir à comparer deux chaînes par le nombre de caractères différents.

III-2.3.3. *Remarques* :

1) On peut constater que ces requêtes étendues impliquent la définition d'un préordre, d'un ordre ou d'une distance, ou encore toute autre propriété relationnelle ou topologique entre objets : cette définition ne pourra naturellement être le fait que d'une implémentation particulière.

2) Dans les deux cas, les requêtes sur les objets se décomposent en propriétés des attributs. Ainsi, par exemple, la distance euclidienne s'élabore-t-elle en 2 étapes :

. valeur absolue de la différence des 2 attributs élevée au carré

. sommation des termes précédents puis calcul de la racine carrée.

3) Plus généralement, toutes les requêtes évoquées jusqu'ici sont ainsi décomposables ; à la limite, toutes peuvent être ramenées à une **combinaison booléenne** de requêtes à attribut unique. Suivant la complexité relative de la requête, celle-ci sera traitée comme une **succession de requêtes** plus simples, elles-mêmes éventuellement combinaisons logiques de propriétés sur plusieurs attributs (c'est-à-dire pratiquement des requêtes d'identité partielle).

Cette propriété est extrêmement intéressante : elle permet en effet une prise en charge par le matériel d'un niveau plus ou moins puissant de l'algorithme suivant les possibilités câblées (ou micro-programmées) disponibles. Elle interagit fortement avec l'architecture même du système car la décomposition d'une requête se traduira, soit par un éclatement parallèle dans un contexte multiprocesseur , soit par une accumulation séquentielle dans le cas d'un monoprocesseur (où l'on pourra chercher à réduire l'espace mémoire exploré successivement). Ce point sera explicité plus loin.

4) On pourrait donc considérer que les requêtes précédentes ont la particularité de pouvoir se définir par **intersection** de requêtes élémentaires, c'est-à-dire par l'**inclusion successive** des sous-ensembles fournis par ces requêtes élémentaires.

En outre, les types de requêtes présentés ne sont pas les seuls qu'on puisse imaginer ; ils ne constituent qu'une certaine caractérisation de ces requêtes, mais en incorporant celles-ci à des algorithmes appropriés, il est possible d'en définir d'autres, adaptées aux besoins propres à l'utilisation.

5) A ce qui précède, ajoutons enfin que la ressemblance avec seuil fait quelque peu exception. Lorsqu'on procède attribut par attribut, on ne peut qu'éliminer les objets tels que d partielle (A, C_i) devient supérieure au seuil : ces objets constituent petit à petit $C_M(Q_i(F))$, le complément de $Q_i(F)$. Il est donc possible de décomposer Q_i en $Q_{i1} \ldots Q_{im}$ de sorte que $Q_i(F) \subset \bigcap_{h=1}^{m} Q_{ih}$.
Cependant la propriété caractéristique de $Q_{ih}(F)$ n'est pas purement locale : le seuil auquel on confronte le résultat partiel est lié au résultat global du calcul de la distance. Autrement dit, il n'existe pas de seuil local permettant d'éliminer sur cette seule évaluation une partie de l'espace à explorer. Ainsi, il se peut que des éléments de $Q_i(F)$ soient tels qu'il existe des a_j pour lesquels $d(c_i, a_j)$ ne soit pas minimale. Les requêtes élémentaires issues de la décomposition doivent donc comporter le même seuil que Q_i.

Prenons un exemple :

soit F = {AIMERA, AIMANT, LIMACE, REVERA , LIMEUR, RIDERA, RAMEUR}

si on suppose que d(A, A') est le nombre de lettres différentes (dans des positions correspondantes)

on peut définir $Q_i(C, \lambda)$ sur cette distance.

Soit C = (LIMERA)

$\lambda = 2$

alors $Q_i(F)$ = {AIMERA, RIDERA, LIMEUR}

Si on décompose Q_i en Q'_i et Q''_i

tels que Q'_i = (L I M ▪ ▪ ▪, 2) et Q''_i = (▪ ▪ ▪ E R A, 2)

alors $Q'_i(F)$ = {AIMERA, AIMANT, LIMACE, LIMEUR, RIDERA, RAMEUR}

et $Q''_i(F)$ = {AIMERA, REVERA, RIDERA, LIMEUR, RAMEUR}

on a forcément : $Q_i(F) \subset Q'_i(F) \cap Q''_i(F)$

et en outre $Q_i(F) = Q_i(Q'_i(F))$

$= Q_i(Q''_i(F))$

III-2.3.4. *Requêtes globales*

On appellera ainsi des requêtes telles que leur traitement implique, sur M :

- l'existence d'un **préordre total**

et/ou - la définition partout d'une **distance** d(A, A').

Autrement dit, contrairement aux cas précédents, $Q_i(F)$ dépend de F tout entier et non de propriétés purement locales des objets. Sans pouvoir dresser un catalogue complet de telles requêtes, on peut envisager deux exemples particulièrement intéressants :

1) Meilleure ressemblance

La requête de meilleure ressemblance est particulièrement importante ; elle permet en effet de supporter des tâches de type "reconnaissance de formes". On peut, sous l'angle de la théorie de l'information, l'assimiler à l'usage d'un **code correcteur** permettant d'attribuer un comparande inconnu (et souvent bruité) à une classe d'équivalence à l'aide d'une règle de décodage appropriée.

On peut définir une telle requête de la façon suivante :

$$R_i(F) = \{A \in F, \neg(\exists A' \in F), d(A', C_i) < d(A, C_i)\}$$

c'est-à-dire que $R_i(F)$ contient les objets A dont la distance au comparande C_i est minimale.

Elle présente quelque différence avec la "ressemblance avec seuil" (III-2.3.2. 2)).

Elle n'est en effet pas décomposable en requêtes successives sur des sous-ensembles inclus dont chacun contient $R_i(F)$. Ainsi, si nous reprenons l'exemple de III-2.3.3. 5), avec le même F et la même distance d, R_i étant défini maintenant comme une requête de meilleure ressemblance, on aura par exemple :

$R_i(F, \text{LIMERA}) = \{\text{AIMERA}\}$

$R_i(F, \text{HUMEUR}) = \{\text{LIMEUR}, \text{RAMEUR}\}$.

Contrairement au cas de la requête avec seuil, il n'est pas possible de trouver une requête partielle (par rapport au nombre d'attributs, i.e. ici, de lettres) telle que $Q_i(F)$ soit contenu obligatoirement dans $Q'_i(F)$.

On peut exprimer cette notion de façon plus intuitive en disant qu'on ne peut savoir a priori sur quels attributs se fera la meilleure ressemblance. Au contraire, dans le cas de la référence à un seuil, on est certain de pouvoir éliminer les objets pour lesquels on dépasse le seuil avant la fin de l'évaluation.

Toutefois, dans un contexte **multitraitement** on notera que la requête de meilleure ressemblance peut être décomposée en :

- une phase parallèle de recherche de la meilleure ressemblance dans des sous-ensembles $F_1 \ldots F_M$ de F tels que $\bigcup_{j=1}^{M} F_j = F$. On obtient alors des $R_i(F_j)$ $(1 \leq j \leq M)$
- une phase globale de concertation dans laquelle on effectue Q_i sur les réponses obtenues précédemment, soit $R_i(\bigcup_{j=1}^{M} R_i(F_j))$.

En fait, on rejoint dans cette approche le problème du tri puisque l'on peut assimiler la requête de meilleure ressemblance à un tri (ou plutôt

à une recherche de minimum) dans l'espace des distances $d(C_i, A)$ $\forall A \in M$.

Une autre modification possible, en pratique, de cette présentation de la meilleure ressemblance est liée aux propriétés des objets manipulés eux-mêmes. En effet, il est en général possible de définir un seuil au delà duquel la ressemblance sera invraisemblable ou peu significative (en termes de probabilités). Outre le fait de pouvoir dès lors abandonner un possible meilleur candidat en cours de recherche, on pourra éventuellement combiner les deux types de requêtes de ressemblance. Ainsi, en reprenant toujours le même exemple, soit
Q_i la requête définie en III-2.3.3., 5)
$Q_i(F)$ la réponse à cette requête
soit R_i la requête de meilleure ressemblance.
Si on estime, grâce à des considérations (d'ordre statistique par exemple) liées au problème traité que le meilleur voisin sera contenu dans $Q_i(F)$ (dont le seuil rappelons le est $\lambda = 2$) alors, on fera : $R_i(Q_i(F))$ au lieu de $R_i(F)$. Et on a bien par exemple

$$R_i(Q_i(F, \text{LIMERA})) = R_i(F, \text{LIMERA})$$

Remarquons que cette attitude est à rapprocher du cas de la théorie du codage. En effet, les codes correcteurs utilisés sont capables de corriger un nombre fini d'erreurs, selon leur redondance. Le choix de celle-ci dépend alors de la probabilité d'erreur et de compromis entre coût d'une erreur non corrigée / coût de la correction, déterminés grâce à la connaissance du procédé de transmission, de stockage, etc ...

Dans certains cas, on pourra appliquer cette idée directement à la reconnaissance de formes. Ainsi, pour la reconnaissance de caractères manuscrits, ICHIKAWA et al. [110] ont-ils adopté une telle technique. Le codage des formes de caractères lus est tel que l'application d'une règle de décodage permet d'attribuer une forme inconnue à un mot-code représentant une forme de référence. De ce fait, la probabilité d'une erreur de détection (forme d'entrée attribuée à une mauvaise classe) est directement liée au nombre de bits du code que l'on peut corriger au décodage.

Une telle méthode suppose donc un choix très délicat de la fonction de codage, et devient inapplicable dans le cas de formes plus élaborées c'est-à-dire beaucoup plus difficiles à paramétrer et donc à doter d'un code correcteur suffisamment redondant. Plus précisément, ce n'est pas vraiment la redondance de l'information qui pose problème mais la **fonction de décodage** permettant d'extraire le mot-code original. Un cas également assez problématique sera celui où le choix des paramètres pertinents reste relativement empirique, lorsque le phénomène observé manque d'une connaissance théorique suffisamment poussée : il sera alors nécessaire, faute de savoir définir exactement les **attributs** qui caractérisent une forme, de conserver le plus possible d'informations brutes.

En tout cas, la requête de meilleure ressemblance est, par excellence, la fonction associative qui relève du type d'informations traitées (voire du type de traitement effectué sur ces données) : en effet, la définition même de la distance est liée étroitement à ces contraintes. Une machine capable de "mesurer" la ressemblance sera donc nécessairement spécialisée, tout au moins s'il y a lieu de respecter des limites en coût matériel et en temps de réponse.

2) Recherche d'extrema

Une autre façon d'exprimer les relations qui prennent en compte tout F est de définir sur F ou sur M une relation d'ordre ou, seulement, un **préordre**.

Cette relation est autrement puissante que dans le cas de la requête d'identité partielle étendue (III-2.3.2. 1)) où l'on admettait des inégalités ou des préordres entre attributs : Ici, il s'agira de définir une relation de préordre **entre objets**.

Deux types de requêtes peuvent alors être envisagées :

- l'une, assimilable aux requêtes étendues, consiste à rechercher le sous-ensemble des objets inférieurs (ou supérieurs) à un comparande donné. On a alors :

$Q_i(F) = \{A \in F,\ A\ \alpha\ C_i\}$

{où α dénote la relation de préordre}

Une particularité intéressante de ce genre de requête est qu'il est possible d'optimiser leur traitement en utilisant une stratégie basée sur le préordre. Par exemple, ce préordre étant supposé connu il sera intéressant de procéder à une organisation du stockage qui en tienne compte.

- l'autre, plus globale, consiste à rechercher véritablement les extrema (notons qu'en cas de relation d'ordre chaque extremum sera unique). On aura : (si α est le préordre) :

$$Q_i(F) = \{A \in F, \neg(\exists A' \in F), A' \alpha A\}$$

Il sera ainsi possible de définir, pour un préordre donné, deux sous-ensembles MIN et MAX.

Sans détailler davantage, remarquons qu'il existe une certaine analogie entre cette requête et la "meilleure ressemblance" : à la limite, en associant une relation de préordre aux valeurs des distances, celle-ci peut être considérée comme un cas particulier de la recherche d'extrema : un objet appartenant à $R_i(F)$, sous-ensemble des plus proches voisins de C_i, peut également être vu comme faisant partie du MIN(F) pour les distances.

III-2.3.5. *Problème des objets de longueur variable*

Jusqu'ici, nous avons implicitement supposé que le nombre d'attributs que comporte un objet est fixe ; en fait, cela correspond au cas où les objets sont de taille fixe. A condition d'imposer certaines restrictions, nombre de problèmes peuvent être traités de cette façon.

Cependant, dans certains cas, il n'est pas possible de normaliser la taille des objets. Ce sera notamment le cas lorsque le procédé d'acquisition des données implique la variabilité du nombre des attributs. Ainsi, dans les contextes de la transmission de données ou du traitement du signal peuvent survenir des erreurs (du bruit) par insertion ou omission d'attributs. Cela se traduit au niveau des objets par la dilatation et/ou la contraction d'une dimension liée au nombre d'attributs (généralement le temps).

. Dans le cas des requêtes partielles, certaines précautions peuvent permettre de tenir compte de ce phénomène :

- on peut considérer comme un attribut le nombre d'attributs (longueur) : les requêtes pourront préciser sa valeur ou des valeurs extrêmes.En outre la possibilité de masquer cet attribut "longueur" permet de traiter des requêtes partielles où seuls les attributs souhaités seront précisés : ainsi, on recherchera $Q_i(F)$ telle que les objets comprennent $a_2 = r_2$, $a_4 = r_4$, $a_7 = r_7$, ... sans exiger que les attributs soient en nombre identique.

- Les requêtes de ressemblance avec seuil peuvent également être traitées dans ce sens. Si la distance définie consiste à comporter un certain nombre d'attributs identiques au comparande (où même, compris dans un intervalle donné) il n'est pas nécessaire que les longueurs des objets soient identiques. Ainsi, en reprenant le sous-ensemble F de III-2.3.3. 5), et en définissant d et λ comme "posséder au moins 5 lettres (attributs) identiques" cherchons $Q_i(F)$, où C_i = (REVERIE).

On voit que C_i possède un attribut de plus que les éléments de F. Il existe cependant :

$$Q_i(F) = \{REVERA\}$$

. Les problèmes de reconnaissance, généralement, présentent cette difficulté (notamment la reconnaissance de la parole, nous le verrons plus loin). La définition d'une distance globale entre objets s'avérant alors plus délicate que dans la situation précédente, il faut pouvoir définir des requêtes de meilleure ressemblance qui puissent prendre en compte cette complication. Deux approches peuvent alors être envisagées :

1) Evaluer la ressemblance grâce à des algorithmes itératifs indépendants de la dimension variable des objets : on utilise alors essentiellement des processus de **comparaison dynamique** (figure 19). Leur principe (nous aurons plus loin l'occasion de donner plus de détails) est d'optimiser pas à pas la progression dans un plan (forme inconnue X objet de référence) en calculant en chaque point un indice local et un score partiel pour le chemin considéré. En fait, cela revient à évaluer si deux attributs successifs sont suffisamment proches au sens de la ressemblance pour être confondus (cas d'une insertion erronée) ou si, compte tenu des indices successifs de ressemblance avec la référence, il n'y a pas une probabilité d'omission.

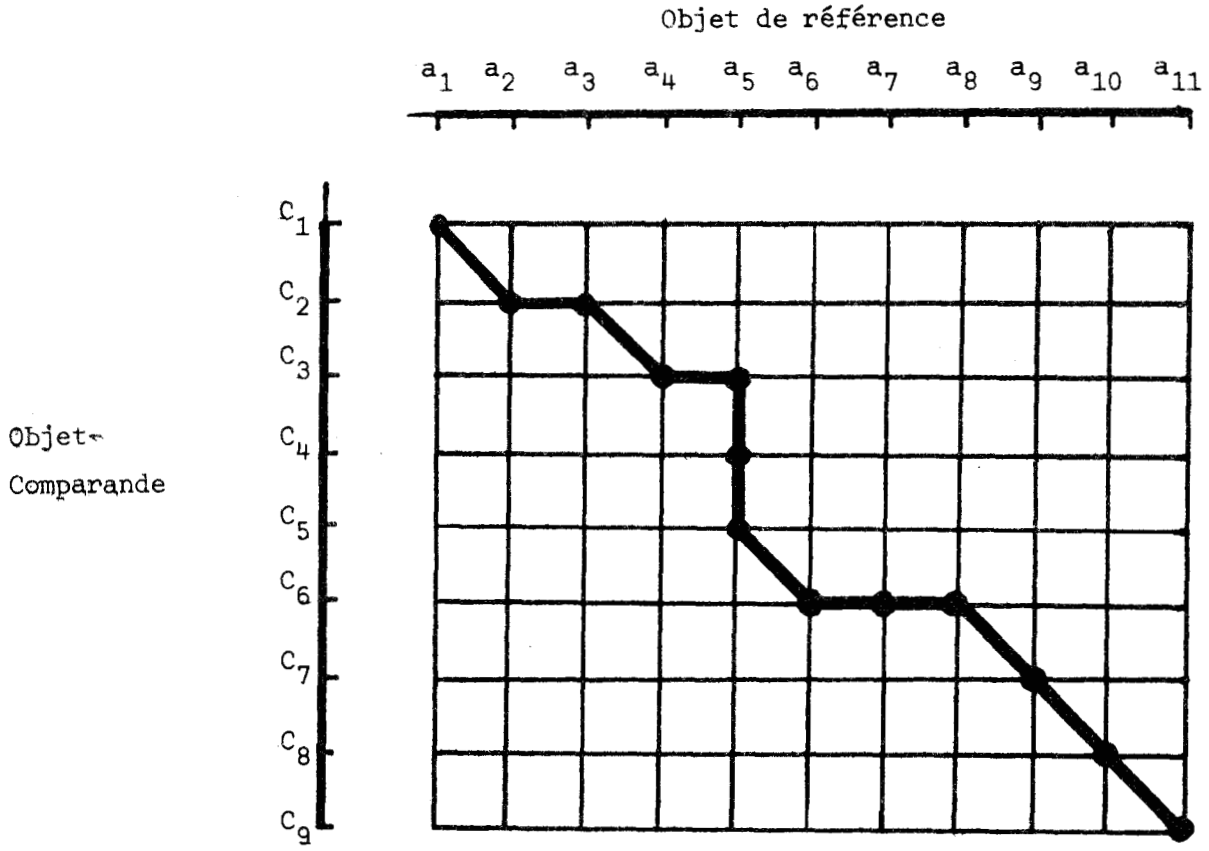


Fig. 19 : Comparaison dynamique (indépendante de la longueur)

Le principe est de définir une relation de récurrence permettant de calculer itérativement un score pour chaque chemin, sommet par sommet. Le "meilleur" chemin est celui pour lequel le score obtenu est optimal.

On détecte ainsi dans le comparande les probabilités d'erreurs d'acquisition ; par exemple, ici :

- omission on passe, dans C, de C_6 à C_7 : a_7 et a_8 sont "omis"
- insertion C_3, C_4 et C_5 sont des "répétitions" de a_5

N.B. Il semble, intuitivement, que ce processus de comparaison est d'autant plus fiable que $\frac{\Delta L}{L}$ reste relativement faible : c'est notamment le cas si le nombre d'attributs est suffisamment grand. D'autre part, comme les 2 types d'erreurs s'annulent du point de vue de la longueur, il faut les prendre en compte de façon pénalisante dans le calcul du score.

L'avantage de ces méthodes est de travailler directement sur l'information acquise, et aussi de fournir un résultat relativement fiable. Leur inconvénient majeur est de se révéler coûteuses en puissance (temps X espace mémoire) de calcul : en effet, pour un objet à reconnaître, il faut évaluer tous les chemins de coïncidence avec chaque objet de référence.

2) Une autre option consiste alors à extraire des objets (comparandes et références) des propriétés indépendantes de leur longueur. On substitue alors, aux attributs "bruts" des objets, d'autres attributs intermédiaires qui peuvent être, par exemple :

- moyennes des attributs
- centre de gravité
- "relief" (valeur rendant compte des écarts maximaux)
- etc ...

Tout se passe comme si l'on effectuait une transformation ou un transcodage d'une représentation dans une autre. La difficulté essentielle est ici de déterminer cette transformation de façon à assurer une "conservation" de la ressemblance au cours de cette transformation. Autrement dit, il faut que globalement deux objets distincts dans la représentation brute le restent après extraction des paramètres et aussi qu'une ressemblance "brute" se traduise dans les attributs extraits. Le choix des transformations à effectuer est donc très délicat et sera souvent assez empirique. On peut considérer qu'il correspond à une phase expérimentale, lorsque les données traitées sont mal connues. Ensuite, le passage à une représentation des objets par un nombre fixe d'attributs permet d'effectuer normalement les requêtes associatives.

III-3 - REQUÊTES ASSOCIATIVES

III-3.1. Influence du nombre d'objets requis et du nombre d'attributs à examiner

Dans ce qui précède, on a essentiellement caractérisé les requêtes d'après la définition de sous-ensembles à partir des attributs.

Or, s'il est intéressant de pouvoir définir des sous-ensembles d'objets à court terme de façon à effectuer des interrogations complexes il convient de distinguer deux types de manipulations de la mémoire associative : en effet, pour une requête Q_i, il est tout à fait différent de chercher $Q_i(F)$ et de chercher "des éléments (ou un seul) de $Q_i(F)$".

La définition de $Q_i(F)$ se traduira le plus souvent par un marquage des objets concernés, ce qui permettra ensuite en interrogeant sur cette marque de ne travailler que sur $Q_i(F)$. Le plus souvent en raison du nombre conséquent d'objets marqués, il ne serait pas rationnel d'effectuer une lecture complète de $Q_i(F)$. Au contraire, lorsqu'on souhaitera lire des objets (vers l'extérieur ou une autre partie du système) ceux-ci seront en petit nombre.

Du point de vue de certaines requêtes, il est évident qu'une interrogation du type "rechercher trois éléments de $Q_i(F)$" aura un temps de réponse moyen probablement plus faible que "rechercher $Q_i(F)$" : en effet, dans ce dernier cas il sera nécessaire d'examiner la totalité de la mémoire.

De même, les requêtes globales ne sont pas, contrairement aux autres (identité d'attributs et requêtes étendues), décomposables en requêtes successives plus simples qui permettraient de réduire fortement le nombre d'accès nécessaires.

En effet, en raison du coût (en temps notamment) des accès à la mémoire, il est intéressant de pouvoir économiser le nombre de ces accès.

Par exemple, une requête d'identité partielle sur 3 attributs étant décomposée en 3 requêtes (Q_1, Q_2, Q_3) à attribut unique, on pourra la traiter de la façon suivante :

- recherche de $G = Q_1(F)$, puis
- recherche de $H = Q_2(G)$, enfin
- recherche de $K = Q_3(H)$

on a bien $K = (Q_1 \cap Q_2 \cap Q_3)\ (F)$.

Il faut évidemment disposer d'un mécanisme permettant de n'interroger, pour Q_2, que G, et, pour Q_3, que H. Ce mécanisme, qui introduit souvent un matériel supplémentaire, devra être justifié par le gain en temps de réponse obtenu. Notons également que ce genre d'amélioration se justifiera d'autant plus que Card (K) << Card (H) << Card (G) << Card (F) et que le système possède un degré poussé de séquentialité. En effet, il est probable qu'avec un degré suffisant de parallélisme au niveau matériel, il devienne plus économique d'effectuer directement $Q(F) = (Q_1 \cap Q_2 \cap Q_3)\ (F)$.

Un autre exemple de réduction de l'espace-mémoire exploré est le cas de la ressemblance avec seuil. Comme on l'a vu, il sera généralement possible d'entrelacer le calcul de la mesure inter-attributs et de la distance globale (qui sera le plus souvent une sommation de ces mesures). A condition de prévoir de façon appropriée l'organisation de l'accès aux objets, on pourra abandonner au fur et à mesure les objets pour lesquels $d(A,C_i)$ dépasserait le seuil.

Toutes ces économies d'accès (qui supposent aussi la possibilité d'un accès individuel aux attributs) ne sont valables que lorsque les requêtes peuvent être décomposées.

Les requêtes globales et notamment celle de "meilleure ressemblance" impliquent un examen exhaustif des objets de la mémoire. Si l'on examine les objets successivement, le plus proche voisin ne peut être connu qu'après avoir exploré la totalité des objets. Cependant, dans l'hypothèse d'une recherche séquentielle, il est possible d'éliminer avant la fin de l'évaluation d'un objet ceux pour lesquels $d(A,C_i)$ est supérieure au "score" du meilleure candidat courant. (On retrouve ici les analogies

déjà remarquées aves les problèmes de tri, d'autant qu'une recherche de maximum est en fait un tri authentique quoique partiel).

Les remarques précédentes concernaient deux paramètres essentiels de l'interrogation :

- le nombre d'objets requis : c'est-à-dire souhaite-t-on "un ..." "des ...", ou "tous les objets qui vérifient telle ou telle propriété" ? Le temps de réponse dépend partiellement de cette caractéristique.
- le nombre d'objets à examiner : c'est-à-dire peut-on trouver un procédé de réduction progressive de l'espace mémoire à explorer. On a vu qu'en général c'est impossible pour les requêtes globales.

On peut donc définir deux types de méthodes d'interrogation qui influenceront fortement l'organisation du système et les modes d'accès ; nous les appellerons respectivement :
- méthodes exhaustives : lorsqu'il est nécessaire d'explorer toute la mémoire pour satisfaire une requête
- méthodes sélectives : lorsqu'un mécanisme adapté permettra de réduire sensiblement le nombre d'accès, voire de traduire directement en accès une requête.

Les options que ces 2 approches impliquent, d'une part du côté de l'utilisation (type de tâches traité), d'autre part du côté l'implémentation (organisation, accès, architecture, ...), sont très déterminantes. De ce fait, nous considérerons ensuite que ces deux types de méthodes répondent de façon **mutuellement exclusive** à deux types de problèmes différents. Autrement dit, bien qu'on puisse imaginer un problème ou un système acceptant ces deux types de fonctionnement (simultanément ou alternativement), on admettra que cette distinction permet de caractériser de façon nette le problème à traiter.

III-3.2. Interrogation sélective

Il s'agit donc du cas où il est possible d'optimiser le nombre des accès nécessaires à la satisfaction d'une requête. Au moins, pourra-t-on se contenter d'explorer un **sous-ensemble** de l'espace-mémoire, au mieux une fonction appropriée traduira la requête en un **accès unique**.

Il est évident que ce genre de facilités dépend fortement d'une **pré-connaissance** précise

- des **données** à gérer
- des **requêtes** souhaitées.

Il en est ainsi généralement lors de la construction d'une base de données, par exemple : la structure même de celle-ci est conçue en fonction des relations sémantiques des données et des types d'accès prévus a priori.

On retrouve ici l'idée de traduire la typologie des objets sur la topologie de la mémoire (voir III-1.1.1.).

Nous envisagerons plusieurs types de solutions.

Il s'agit de modes d'organisation qui permettent d'exprimer physiquement les propriétés qui servent de clés aux accès. On peut alors distinguer :

- les structures de données, principalement celles qui favorisent l'expression de relations hiérarchisées entre les composantes des objets : ce sont notamment les bases de données relationnelles, les treillis, les réseaux, les arbres, ...Considérons, par exemple, un dictionnaire stocké sous forme d'arborescence. Par "dictionnaire", nous entendons ici un sous-ensemble des mots du vocabulaire d'une langue donnée. Plutôt que de stocker ces mots en vrac, il est intéressant d'utiliser un arbre : celui-ci peut en effet supporter un **ordre** (alphabétique), permet assez facilement les **mises à jour** et procure une certaine **économie de place**. Les accès souhaitables peuvent concerner la validation d'une orthographe ou encore l'obtention de la traduction dans une autre langue des mots recherchés : dans ce dernier cas, une feuille permettrait d'accéder à cette réponse. Notons encore qu'une simple structure linéaire dans l'ordre alphabétique rendrait des services similaires, par accès dichotomique ; cependant les mises à jour seraient plus délicates.

- les techniques de hash-code : dans ce cas, une fonction permet, théoriquement, de traduire en termes d'accès les requêtes. Plusieurs difficultés se posent : d'une part, le choix d'une fonction permettant de réduire les "collisions" (et, corollairement, la gestion de ces dernières),

d'autre part la possibilité de définir une fonction multi-attributs, c'est-à-dire capable de prendre en compte plusieurs attributs simultanément. En général, plutôt que de travailler globalement sur toute la mémoire, on éclate celle-ci en **paquets** dont la taille et la composition sont assez délicates à déterminer : certains objets peuvent alors se retrouver dans plusieurs paquets. Les requêtes s'effectuent alors en deux temps : d'abord on accède à un (dans certains cas, plusieurs) paquet(s). Puis on travaille ensuite dans le (ou les) paquet(s) choisi(s). De ce fait, si la fonction de choix du paquet est suffisamment discriminante et si le nombre des paquets est assez grand, on aura réduit sensiblement l'espace exploré.

Ce genre de techniques est assez bien adapté à des requêtes simples sur des attributs multiples comme c'est le cas notamment de la **gestion de fichiers** (voir notamment RIVEST [77]). En effet, dans ces problèmes, on connaît parfaitement les deux parties du cahier des charges que sont les données et les requêtes à effectuer.

Dans le cas où cette définition **a priori** des données et/ou des requêtes fait défaut, ces techniques deviennent relativement impraticables : il se peut en effet qu'on ne puisse ou ne souhaite figer les accès à des données : une organisation plus souple est alors nécessaire ; inversement, les données peuvent être mal définies ou peu fiables : ainsi, en général, en est-il de la **reconnaissance de formes** : les requêtes sont essentiellement du type "meilleure ressemblance" mais les données sont souvent difficiles à caractériser. On rejoint ici les problèmes classiques de la **classification des données** (notons d'ailleurs l'analogie des techniques de hash-code avec ces derniers) :

- choix d'une fonction d'analyse
- choix d'un représentant par classe
- gestion des redondances éventuelles.

Lorsque les formes à identifier sont relativement faciles à caractériser, cette approche s'avère toutefois possible : ainsi l'utilisation, déjà évoquée (cf. ICHIKAWA [110]) des **codes correcteurs** est une illustration de ces principes : le codage des objets est alors supposé suffisamment redondant pour que la règle de décodage d'erreurs permette de retrouver le code correct, c'est-à-dire la forme à identifier.

Finalement, on vérifie assez bien sur ces exemples que la possibilité de recourir à de telles techniques est essentiellement liée au type de problème traité : les solutions évoquées, en effet, relèvent toutes d'un contexte d'application particulier.

Pour résumer, on peut considérer qu'il est possible de réduire l'espace mémoire soumis à une requête associative s'il existe une ou plusieurs propriétés (relation d'ordre, codage, ...) des objets qui permettent d'organiser le stockage et/ou l'accès de façon à respecter cette propriété. En outre, le type de requête intervient également, on l'a vu, puisqu'une requête de meilleure ressemblance par exemple exige une exploration complète des objets de la mémoire.

Finalement, un certain nombre de tâches de type associatif peuvent se ramener à des méthodes à **interrogation sélective** : il faut pour cela recourir à des techniques telles que les structures de données, ou le hash-code. Bien que parfois délicates à mettre en oeuvre, ces techniques ont fait l'objet d'investigations assez nombreuses. En outre, leurs relations étroites avec la notion **d'accès aléatoire** et l'algorithmique classique les prédisposent à s'appuyer sur des architectures matérielles classiques : en effet, l'adressage dans un monoprocesseur peut parfaitement supporter (dans certains cas il est indispensable) le traitement de telles fonctions. De ce fait, dans ce domaine, les travaux doivent porter essentiellement sur le **logiciel**. Au niveau du matériel, les améliorations apportées sur les machines d'usage général seront bien sûr prises en compte mais de façon indirecte.

III-3.3. Interrogation exhaustive

Dans les cas autres que ceux évoqués ci-dessus, il est nécessaire de parcourir de façon exhaustive tous les objets de la mémoire. Remarquons d'emblée que ce cas correspond de façon plus stricte au traitement associatif tel qu'il a été défini au chapitre I : en effet, la notion de manipulation **en parallèle** des objets était une des conditions admises.

Par rapport aux solutions précédentes, on peut noter deux avantages assez intéressants :

- d'une part, la nécessité d'une interrogation de **tous les objets** de la mémoire implique l'inutilité des fonctions d'adressage au niveau matériel : on respecte ainsi la volonté exprimée dans la description fonctionnelle (voir chapitre II) de considérer la mémoire comme un ensemble d'objets "en vrac".

- d'autre part, l'absence d'une (ou plusieurs) fonctions d'accès topographique (hash-code, arborescence, ...) définies a priori supprime l'aspect statique et figé des accès possibles : le système doit donc y gagner en **souplesse** et en faculté d'adaptation.

La notion de **parallélisme** du traitement au niveau global de l'ensemble des objets pourra se traduire au niveau effectif de l'implémentation de différentes manières. Sans reprendre la discussion des diverses possibilités d'architecture (voir chapitre I) nous insisterons ici sur le fait que le degré plus ou moins poussé de parallélisme du matériel ne peut se justifier que par un taux temps de réponse/capacité plus ou moins exigeant.

Dans bien des cas, une étude suffisamment approfondie montre qu'il est souvent possible de se ramener à des requêtes simples sur un monoprocesseur. L'exemple typique est celui évoqué par HIGBIE [85] de certains systèmes de contrôle de trafic aérien. En fait, dans l'intervalle séparant un processeur associatif totalement parallèle (actuellement irréaliste) d'une recherche séquentielle programmée en mémoire adressable de nombreuses variantes sont possibles. On peut toutefois noter qu'à condition d'être compétitives, des mémoires circulantes représentent une alternative préférable dans la mesure où l'accès aléatoire présente ici peu d'intérêt.

Cependant, dans certains cas, il peut arriver que les performances exigées justifient un degré de parallélisme plus poussé. De ce fait, la conception d'un système à interrogation exhaustive doit englober la définition d'une architecture matérielle adaptée, contrairement aux cas où la solution (logicielle) du problème pouvait s'appuyer sur un matériel courant.

III-4 - INTERROGATION EXHAUSTIVE ET ARCHITECTURES EVOLUÉES

III-4.1. Interrogation exhaustive et multitraitement

On a vu que les interrogations sélectives conviennent aux machines séquentielles à adressage.

De même, certains problèmes nécessitant une **interrogation exhaustive** de la mémoire peuvent néanmoins se satisfaire de machines classiques : en effet, les performances modestes au niveau temps de réponse peuvent permettre d'effectuer un programme séquentiel d'interrogation sans que cela s'avère gênant. Mieux, on pourra, pour respecter les contraintes, combiner les deux types de stratégies. Il en est ainsi, par exemple, de certaines bases de données associatives : le temps de réponse à une interrogation par un opérateur étant sans rapport avec les temps élémentaires au niveau matériel, le matériel reste relativement classique.

Seuls, certains types de problèmes échappent aux solutions précédentes —il semble bien, comme on le verra plus loin, que des tâches de type reconnaissance de formes rencontrent cette difficulté—. Il s'agit des cas où les recours précédents deviennent impraticables :

- les **données** sont **complexes** et se prêtent donc mal à un traitement par les stratégies à interrogation sélective : en effet,celles-ci présupposent une profonde analyse des données traitées de façon à définir les fonctions d'accès nécessaires.

- les **requêtes** sont essentiellement **globales** (par exemple "meilleure ressemblance") et donc impliquent une **interrogation exhaustive.**

- les **performances** souhaitées (notamment, par exemple, un temps de réponse comparable au temps réel) interdisent l'emploi d'un matériel classique, sur lequel une interrogation exhaustive s'avèrerait rédhibitoire.

Le système correspondant, qui sera obligatoirement spécialisé, devra donc associer :

interrogation exhaustive

et **architecture évoluée.**

Par "architecture évoluée", nous entendons ici une structure matérielle telle que le degré de parallélisme introduit se traduise par un gain en temps de réponse appréciable si on la compare à une solution programmée sur un monoprocesseur classique.

On se rappelle que toute implémentation originale d'un processeur associatif ne peut se concevoir que par rapport à une tâche bien définie (cf. chapitre I) : c'est ce que nous envisagerons dans la dernière partie de cette étude. Nous pouvons cependant examiner, sur plusieurs cas typiques, de quelle manière faire face au problème de l'interrogation associative exhaustive.

On peut estimer, a priori, que dans l'hypothèse d'un multi-processeur associatif, si P est le nombre de processeurs capables d'interroger simultanément la mémoire, le **temps de réponse à une requête sera approximativement divisé par p** (par rapport à un monoprocesseur traitant la même requête sur le même nombre d'objets). En réalité, il faudra ajouter pour le multi-processeur les phases de synchronisation : en effet, il faut communiquer les requêtes à tous les processeurs, et, surtout, il faut établir une **concertation** entre ceux-ci pour élaborer un résultat global : nous reviendrons sur ce point.

Rappelons également que, de même qu'en première approche les méthodes sélectives correspondent aux mémoires adressables, il semble intéressant de faire coïncider **interrogation exhaustive** et **accès par le contenu**. A cet égard, des technologies de **mémoires circulantes** représentent une solution attrayante : en effet, outre un coût du stockage théoriquement moindre, la fonction de **scrutation séquentielle** y est implicite. Il reste qu'une implémentation utilisant des mémoires aléatoires (RAM notamment) peut être une solution préférable : dans ce cas, bien que la fonction d'adressage soit inutile pour l'interrogation exhaustive, elle reste disponible,ce qui confère au système d'autres possibilités. De plus, pour des raisons à la fois technologiques et commerciales, les mémoires circulantes, à l'heure actuelle, semblent moins compétitives (cette situation étant toutefois susceptible d'évoluer).

Les configurations que l'on considère sont hypothétiques et générales. Elles correspondent à diverses façons possibles de répartir l'interrogation exhaustive d'une mémoire associative associée à un multiprocesseur. Compte-tenu des remarques précédentes, on suppose que l'espace-mémoire se compose d'une ou plusieurs **séquences de mots**. Selon les cas, une séquence sera bouclée ou non. En fait, on peut énumérer quelques hypothèses sur la façon d'organiser la mémoire (figure 20)

(a) - une séquence unique : ce cas est à rejeter puisqu'il s'agit alors d'un monoprocesseur.

(b) - p séquences disjointes ; on peut considérer que chaque processeur dispose de sa mémoire locale.

(c) - p anneaux disjoints ; ce cas est analogue au précédent : il n'en diffère que par la technologie.

(d) - un anneau unique comportant p fenêtres ; fictivement, chaque processeur gère une mémoire locale mais son contenu change.

Notons que les séquences bouclées, c'est-à-dire les anneaux, outre leur compatibilité évidente avec les technologies de mémoire circulante possèdent un avantage important : dans le dernier cas, en effet, l'anneau sert à la fois d'outil de stockage et d'**organe de communication** entre les processeurs. C'est cette double fonction qu'utilise la machine MAUD (voir, par exemple [111], [112]).

En ce qui concerne le rôle des processeurs dans la gestion des requêtes, on peut imaginer répartir l'interrogation de plusieurs façons. On suppose désormais que les mémoires (locales ou unique) sont des anneaux : en effet, dans le cas d'une séquence de mémoire "ouverte" l'interrogation exhaustive se traduit par une boucle de scrutation (programmée ou séquencée par le matériel) ; on a donc bien une fonction équivalente à celle de l'anneau, la seule différence étant que dans l'anneau matériel l'information circule effectivement.

Le multitraitement des requêtes associatives par interrogation exhaustive peut être envisagé selon l'une des diverses architectures parallèles habituellement évoquées. Nous en examinerons plusieurs.

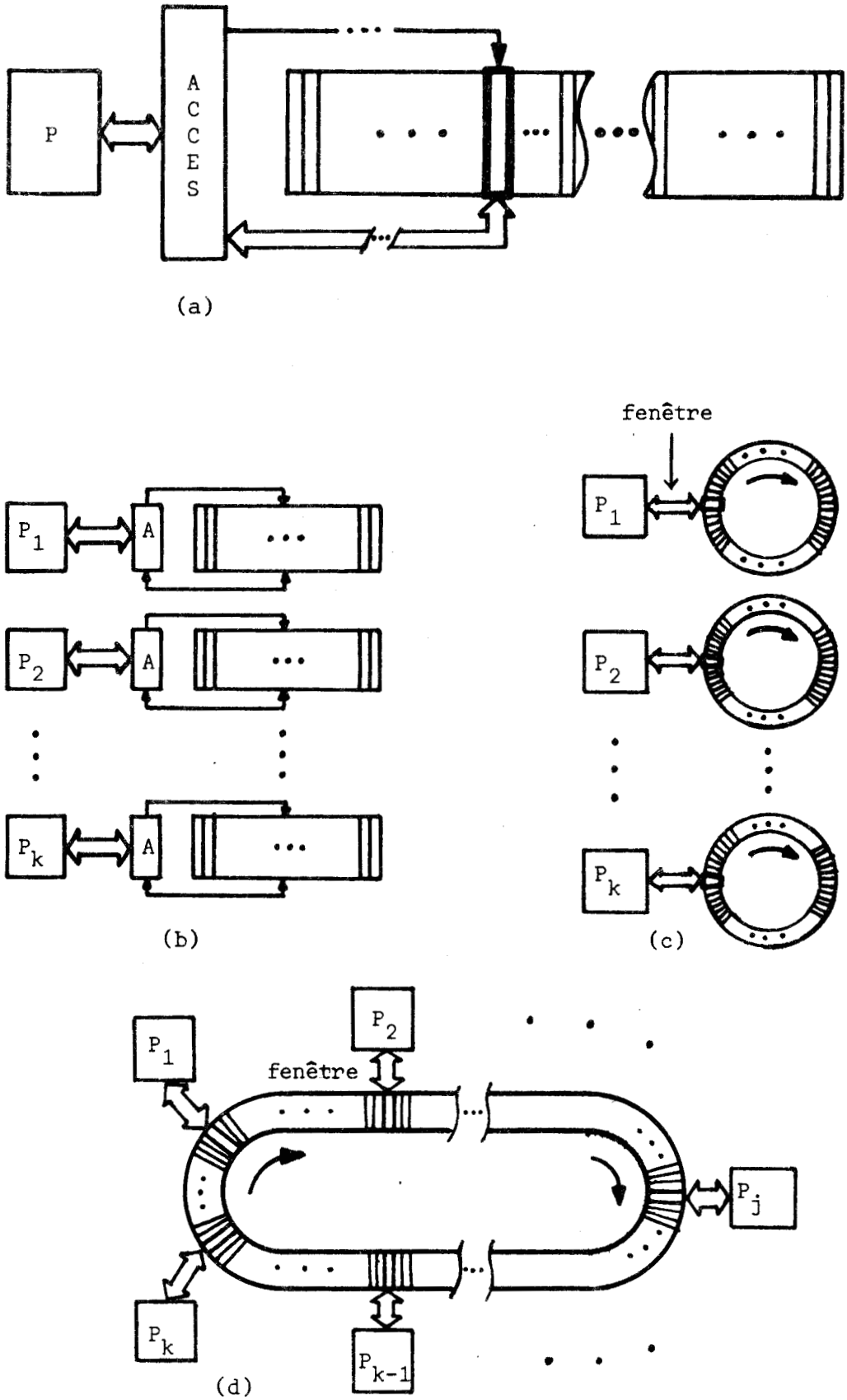


Fig. 20 : Organisations possibles d'une mémoire à interrogation exhaustive

III-4.2. Machine SIMD

Les processeurs d'interrogation sont banalisés et traitent simultanément la même requête sur leur portion d'espace mémoire. Cette dernière peut être locale ou être un secteur d'un anneau unique. La difficulté est d'élaborer un résultat global dans une phase de **synchronisation**. Cette fonction peut être confiée (figure 21) :

- à un processeur de contrôle, qui doit pouvoir recueillir les résultats partiels et effectuer le traitement global.

- à un réseau câblé, si les critères de sélection sont suffisamment simples. Ainsi, on peut imaginer un réseau arborescent analogue aux sélecteurs de réponse multiple (voir I.2.3.) qui puisse valider l'accès au processeur présentant le nombre le plus élevé. Toutefois, cette approche conduit rapidement à une complication matérielle importante.

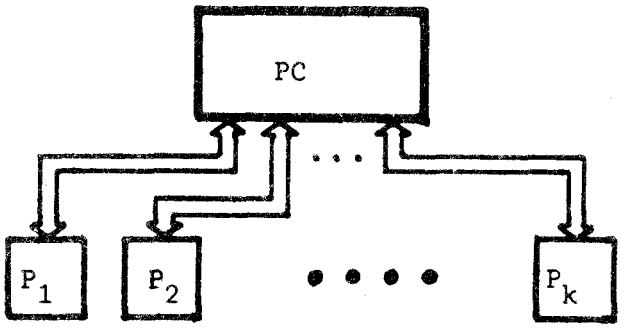
- à un organe de communication permettant aux processeurs eux-mêmes de se concerter. Un exemple simple est celui des communications en cascade (daisy-chain) chaque processeur passant ses résultats à son voisin. L'inconvénient est ici le temps qui est proportionnel au nombre de processeurs.

Le choix entre les deux solutions possibles - anneau unique ou anneaux locaux - ne tient pas aux performances : dans les deux cas, celles-ci seront comparables, à capacité équivalente et nombre égal de processeurs.

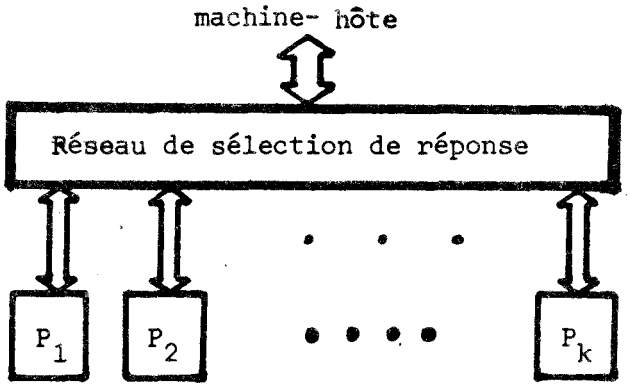
Par contre, ces deux types d'organisation impliquent des caractéristiques différentes sur deux plans :

- d'une part, les **possibilités d'extension** semblent plus faciles dans le cas des processeurs possédant un anneau local de mémoire : il faut pour cela prévoir l'adjonction d'un processeur supplémentaire au système (ce qui peut par ailleur présenter des inconvénients du côté du processus de synchronisation),

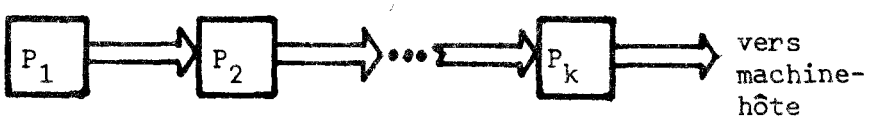
- d'autre part, la **tolérance aux pannes** est différente - une portion de mémoire défectueuse est très gênante dans le cas de l'anneau unique ; par contre celui-ci peut fonctionner en mode dégradé avec un processeur en panne. Au contraire, s'il est possible d'isoler une mémoire



(a) par processeur de contrôle



(b) par réseau câblé



(c) par communication en cascade

Fig. 21 : Synchronisation d'une interrogation parallèle

N.B. (c) peut également représenter, dans le cas d'un anneau unique de stockage, la possibilité d'une communication par l'anneau lui-même.

locale fautive (au prix d'une perte des informations qu'elle contient, c'est-à-dire qu'il faut pouvoir en ré-initialisant le système répartir ce qu'elle contenait sur les autres anneaux), un processeur en panne bloque l'accès aux objets qu'il gérait. Notons en outre que selon la solution adoptée pour la synchronisation ces pannes et leurs conséquences peuvent avoir des effets différents : ainsi, la communication en cascade des processeurs peut être interrompue si l'un d'entre eux doit être isolé (on peut toutefois prévoir la possibilité, dans certains cas, de court-circuiter, dans cette communication, un ou plusieurs processeurs).

III-4.3. Machine "pipe-line"

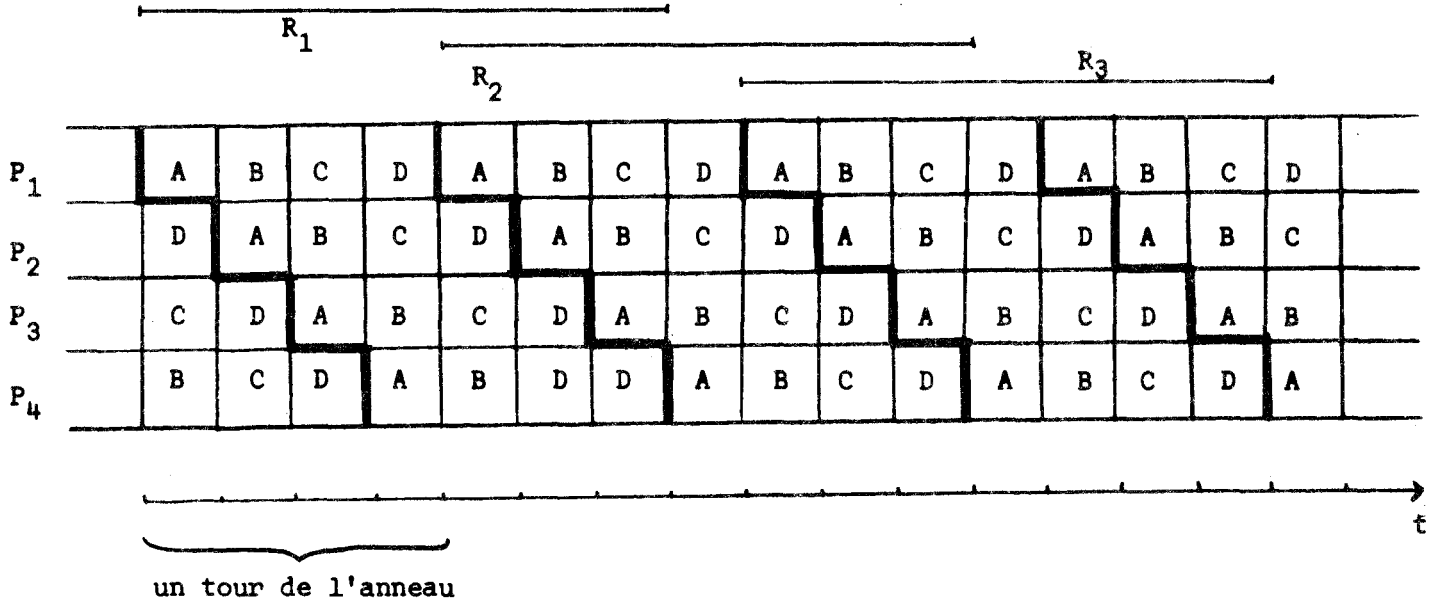
Il s'agit ici de décomposer en plusieurs phases les requêtes. Chaque processeur se voit alors confier l'une de ces phases. On peut alors traiter plusieurs requêtes successives en faisant se chevaucher dans le système les différentes phases de celles-ci. De ce fait, le débit global du système, en nombre de requêtes satisfaites par unité de temps se trouve pratiquement multiplié par le nombre de processeurs - notons cependant que le temps de réponse à une requête individuelle reste identique-.

La difficulté essentielle réside dans la décomposition du traitement d'une requête ; il faut en effet équilibrer les temps exigés par chaque phase de façon à éviter qu'une phase sensiblement plus coûteuse en temps ne puisse retarder l'ensemble du système : en effet, la décomposition des phases est telle que la phase P_i d'une requête doit recevoir les résultats fournis par la phase P_{i-1} pour pouvoir être traitée.

Il paraît par ailleur plus rationnel dans ce cas de choisir la configuration à anneau unique : celui-ci servira à la fois de support de stockage et d'organe de communication entre les processeurs. Il faut prévoir de répartir les processeurs de façon à ce qu'en un tour de l'anneau, une information ait subi toutes les phases successives d'une requête. On pourrait toutefois imaginer de procéder autrement : si le nombre de processeurs est insuffisant pour gérer une requête en un tour, il faudrait alors au bout d'un tour modifier les phases que les processeurs doivent exécuter.

Pour illustrer le cheminement des requêtes dans le système, la figure suivante représente un diagramme des temps. Dans l'hypothèse d'un anneau unique, les phases successives d'une requête doivent impérativement être de durée identique car il faut travailler en synchronisme avec la circulation de l'anneau. On suppose 4 processeurs autour de l'anneau et les requêtes décomposées en 4 phases.

Les phases, en général, doivent être subies dans l'ordre : le processeur P_2 doit donc attendre pour commencer que la première donnée lui parvienne après traitement par P_1, c'est-à-dire un quart de tour qui constitue le décalage entre les phases. On a divisé l'anneau, ou plutôt son contenu, en 4 parties A, B, C, D, et on représente les états successifs de celui-ci par rapport aux 4 processeurs :



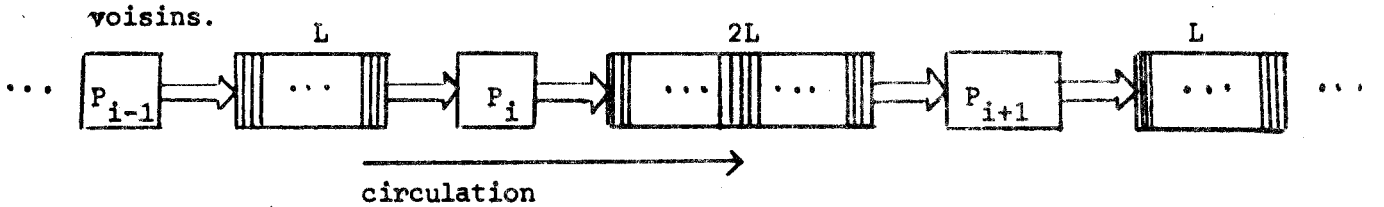
Dans le cas précédent, le débit obtenu n'est pas très avantageux : sur un nombre de requêtes plus grand on verrait qu'il faut $N + \frac{3}{4}$ tours d'anneau pour N requêtes. En outre, le temps de réponse à une requête est médiocre. Cela est dû au chevauchement relativement faible impliqué par la circulation de l'anneau : en fait, un chevauchement meilleur serait obtenu grâce à une communication plus efficace entre processeurs.

Toutefois, l'avantage de cette solution doit être apprécié d'un autre point de vue : la conséquence directe de la décomposition, aussi poussée que nécessaire, d'une requête en un "pipe-line" de "sous-requêtes" est que ces dernières sont nettement moins longues : elles peuvent alors devenir compatibles avec le temps élémentaire de décalage de l'anneau circulant, ce qui n'était peut-être pas le cas de la requête entière. Autrement dit, comme les phases successives de la requête durent environ $\frac{\text{temps de réponse total}}{\text{nombre de phases}}$ il est possible de faire appel à une mémoire circulant plus vite.

Notons encore que la description précédente ne tenait pas compte de certains problèmes, notamment de la phase de fourniture de la réponse, ainsi que de la réinitialisation des objets permettant une nouvelle requête : dans bien des cas il suffira de re-positionner un indicateur.

Une variante possible pourrait survenir dans le cas où certaines phases n'ont pas un ordre strict ; on peut alors améliorer le degré de chevauchement, puisque pour une telle requête, plusieurs processeurs peuvent démarrer en même temps. L'exemple typique est la décomposition d'une requête à k attributs en k requêtes à un attribut. Chaque processeur marquera dans le bloc qu'il examine les objets répondant à l'attribut dont il est responsable lors du premier $\frac{1}{k^e}$ de tour. Puis il suffira de restreindre l'examen pendant les $\frac{1}{k^e}$ de tour suivants aux seuls objets déjà marqués.

La possibilité de décomposer les requêtes en phases de durée égale peut s'avérer une hypothèse trop contraignante. Dans le contexte d'une machine très spécialisée, on peut imaginer d'adapter la longueur de la portion d'anneau séparant 2 processeurs en fonction des durées respectives des sous-tâches dont ceux-ci ont la charge. Cependant une telle faculté reste limitée à des cas de figure relativement simples : ainsi, ci-dessous le processeur P_i a un débit double de celui de ses voisins.



Une autre manière, plus macroscopique, de recourir à un recouvrement des requêtes serait de confier une requête à chaque processeur : il faut alors que chaque objet mémorisé comporte autant d'indicateurs que de processeurs, de façon à pouvoir gérer simultanément (par rapport à l'anneau) plusieurs requêtes.

III-4.4. Machines MIMD

On peut concevoir des architectures de système similaires aux précédentes mais où les processeurs travaillent de façon indépendante. En fait, la distinction que nous introduisons ici est assez arbitraire il faudrait en effet préciser à quel niveau on observe les flux de données et d'instructions (certains auteurs admettent par exemple que les architectures "pipe-line" sont des machines MIMD). Nous envisagerons, car ces machines peuvent revêtir des aspects très divers, deux cas :

1) Dans un contexte de machine analogue à celles décrites en III-4.2, on peut envisager une répartition différente de la prise en charge des requêtes. Celle-ci suppose également une répartition différente du stockage des objets. Au lieu de prévoir des processeurs banalisés gérant chacun un sous-ensemble des objets de la mémoire il s'agit d'une solution "orthogonale" : les processeurs sont spécialisés dans le sens où chacun gère un attribut de tous les objets. Ainsi les objets ne sont plus stockés avec leurs attributs disposés de manière contigue, mais chaque objet est éclaté avec un attribut dans chaque mémoire locale. Cette solution, a priori, serait difficilement compatible avec l'idée de longueur variable des objets. En outre, utilisée de cette façon, elle aurait l'inconvénient de réduire le degré de parallélisme effectif en fonctionnement car une requête décomposée en tâches sur chaque attribut ne mettrait pas à contribution tous les processeurs. Enfin, en raison des dissemblances possibles entre attributs, les sous-requêtes peuvent être de durée extrêmement variable, ce qui complique ensuite la tâche de synchronisation.

2) Par contre, la même idée de confier un attribut à chaque processeur pourrait être exploitée avec profit dans une architecture de type "pipe-line". En effet, si le temps de circulation souhaité est trop élevé pour qu'un processeur, durant son accès à la fenêtre dont il est responsable, puisse évaluer l'ensemble de la requête, on peut alors réduire

le temps nécessaire : il suffit pour cela de ne confier à chaque processeur que l'examen d'un seul attribut. Une nouvelle difficulté surgit alors : il faut pouvoir prendre en charge les requêtes globales, qui ne dépendent pas seulement de propriétés locales aux attributs. Deux types de solutions peuvent alors être envisagées :

. une phase de synchronisation, ou un outil de communication spécifique permettant aux processeurs d'évaluer un résultat global. Cependant, ce serait assez lourd et coûteux en temps de traitement. De plus, chaque processeur devrait gérer une mémoire spéciale accumulant les résultats partiels, pour chaque objet.

. une **accumulation progressive des résultats** : chaque processeur fournit sous forme simple (score ou même indicateur binaire) le résultat partiel de son évaluation. Par exemple, dans le cas de la meilleure ressemblance, chaque processeur fournirait le résultat de l'évaluation des distances inter-attributs entre le comparande et les objets. L'accumulation pourrait se faire dans le support de stockage lui-même (de cette façon, un processeur peut éventuellement décider l'abandon d'un objet si le score partiel dépasse un seuil) ; ce serait assez indiqué dans le cas de l'anneau. Une autre possibilité serait de déposer ces résultats au fur et à mesure dans une mémoire commune gérée par un processeur de contrôle.

La seule difficulté de ce genre d'approche réside dans l'organisation du stockage qui nécessite un éclatement des objets dans les mémoires locales. Ainsi dans un anneau chargé de gérer N objets à k attributs, chaque $1/k^{ème}$ de l'anneau contient N attributs de rang k. Il est alors plus efficace de pouvoir travailler avec k fenêtres, c'est-à-dire la faculté de traiter les k attributs en $1/k^{ème}$ de tour. Toutefois, chaque processeur pourrait très bien examiner, pour une requête donnée, plusieurs attributs. Remarquons également que dans le cas de l'anneau unique, les attributs gérés par un processeur changent alors qu'ils restent identiques dans le cas des anneaux locaux.

Les deux approches précédentes correspondent à des décompositions "intra-requêtes" du traitement, c'est-à-dire à un parallélisme à un niveau assez fin (ce qui implique un **couplage étroit** entre les sous-tâches obtenues). Le traitement peut également être parallélisé à un niveau "supra-requêtes" où on aura alors un **couplage faible**. Le multi-traitement de type MIMD consiste alors à confier des requêtes indépendantes complètes à chaque

processeur. Cette approche convient assez au cas de l'anneau unique dans la mesure où chaque processeur voit défiler la totalité des objets de la mémoire. Là encore, il s'agira de prévoir une multi-indication des résultats, car, pour chaque objet, il faut pouvoir conserver l'indication du résultat de l'évaluation menée indépendamment par chaque processeur. Les problèmes essentiels qui se posent concernent alors la difficulté de rendre les requêtes indépendantes : le résultat de l'une peut en effet conditionner les autres; il faut donc entrelacer sur le système des requêtes dont les résultats sont indépendants, ce qui n'est pas toujours possible. La question de l'intégrité des données que soulève généralement ce genre d'architecture ne se pose pas vraiment, dans la mesure où les requêtes sont essentiellement du type consultation.

CHAPITRE IV

TRAITEMENT ASSOCIATIF DE LA PAROLE

La nécessité est clairement apparue, dans ce qui précède, de situer la conception d'un système associatif dans un contexte précis.

De nombreuses tâches associatives, on l'a vu, peuvent se ramener à des algorithmes implémentés sur des machines classiques. Les seuls cas présentant un intérêt certain sont ceux où il est indispensable d'associer le multitraitement à une méthode d'interrogation exhaustive (III-4.1.). Les caractéristiques des problèmes de reconnaissance de formes, notamment, les rangent dans cette catégorie, surtout lorsque le temps réel s'avère nécessaire.

Ces deux raisons conduisent en particulier à considérer la reconnaissance de la parole comme un terrain favorable à cette approche : en fait, a priori, on peut assimiler la reconnaissance de la parole aux requêtes de meilleure ressemblance. En raison de l'importance, en tant qu'outil de la communication homme-machine, de ce domaine, l'intérêt qu'on peut lui porter semble justifié : comme on le verra, le manque de matériel adapté est d'ailleurs une de ses difficultés essentielles.

Après le rappel des points les plus cruciaux et un bref panorama de la recherche en ce domaine, on examine de quelle manière on peut concevoir l'intervention du traitement associatif dans la reconnaissance automatique de la parole.

IV-1 - LE PROBLÈME DU TRAITEMENT DE LA PAROLE

Il n'est pas question, ici, d'aborder en détail un domaine aussi vaste. Un précédent travail [113] a d'ailleurs permis une approche générale du problème. Nous nous contenterons donc, outre la possibilité d'adopter un point de vue plus récent, de dégager les points essentiels qui caractérisent le problème. Nous donnons, en outre, quelques références bibliographiques récentes ou fondamentales permettant éventuellement de compléter ce qui suit ([114] à [118]).

Il convient ensuite de noter que nous nous intéresserons plus particulièrement à la **reconnaissance** de la parole. En effet, d'une part la synthèse de la parole semble poser moins de difficultés (encore qu'il faille se garder de considérer le problème comme résolu), d'autre part, la reconnaissance s'adapte plus étroitement au genre d'hypothèses considérées en III-4.1. De plus, on considère généralement que les progrès effectués du côté de la reconnaissance ont un impact sur la synthèse, la réciproque n'étant pas vraie : en effet, la différence essentielle tient au fait que dans le cas de la synthèse l'auditeur humain est capable de s'affranchir des imperfections du système.

Dans ce qui suit, on caractérise la tâche de reconnaissance de la parole par les deux pôles qui la composent :
. les données : c'est-à-dire le **signal vocal** et les méthodes permettant de le paramétrer,
. les algorithmes : c'est-à-dire l'imbrication des processus dans un **processus de reconnaissance.**

IV-1.1. Caractéristiques du signal vocal

Le message porté par le signal vocal revêt des formes multiples suivant le niveau auquel on le considère : en effet, les idées s'organisent en **phrases** respectant une certaine **syntaxe** et utilisant un **vocabulaire** donné ; ce vocabulaire est constitué de **mots** construits sur des unités linguistiques caractéristiques de la langue, les **phonèmes.**

Une machine chargée de décoder le message émis par le locuteur doit travailler en fait sur le **signal acoustique** capté par le transducteur. Nous admettrons ici qu'il y a identité entre ce signal acoustique et le signal vocal.

Les caractéristiques essentielles de ce signal sont alors (figure 22) :

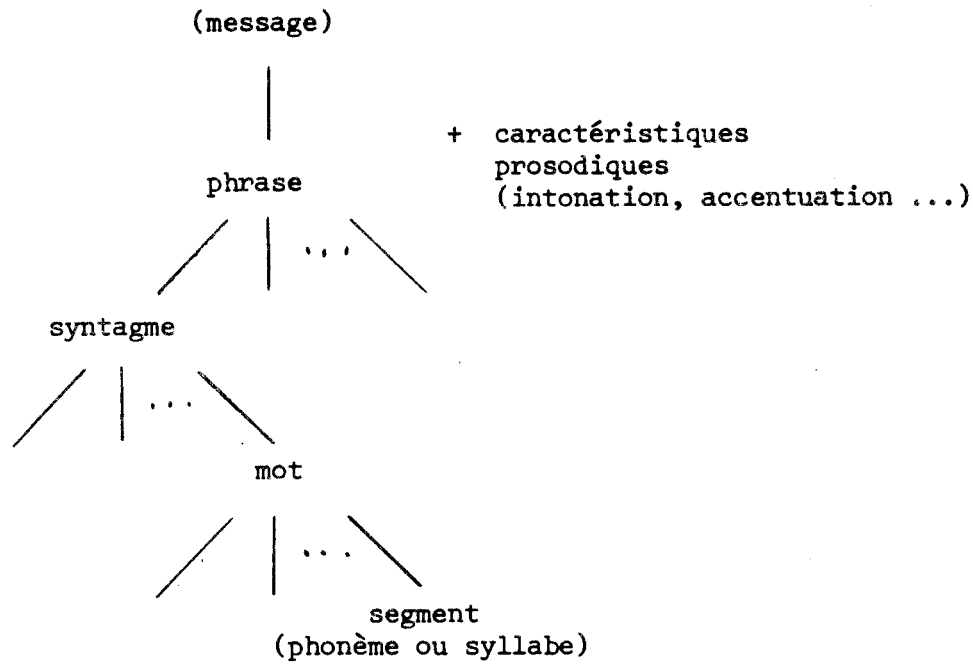
1) Redondance. En tant que source d'information, le signal vocal échantillonné et digitalisé est extrêmement redondant. En effet, son **débit brut**, obtenu d'après le théorème d'échantillonnage est de l'ordre de **50 K bits/s** soit une fréquence d'échantillonnage double de la bande à échantillonner et le nombre de bits utilisés par échantillon (à rapprocher du standard de 64 K bits/s préconisé par le CCITT).

Par contre, le **débit "utile"** du signal serait de **50 à 100 bits/s** : un locuteur prononce en moyenne 5 à 10 phonèmes par seconde et l'on peut définir 30 à 50 phonèmes dans une langue (soit 5 à 6 bits pour les coder).

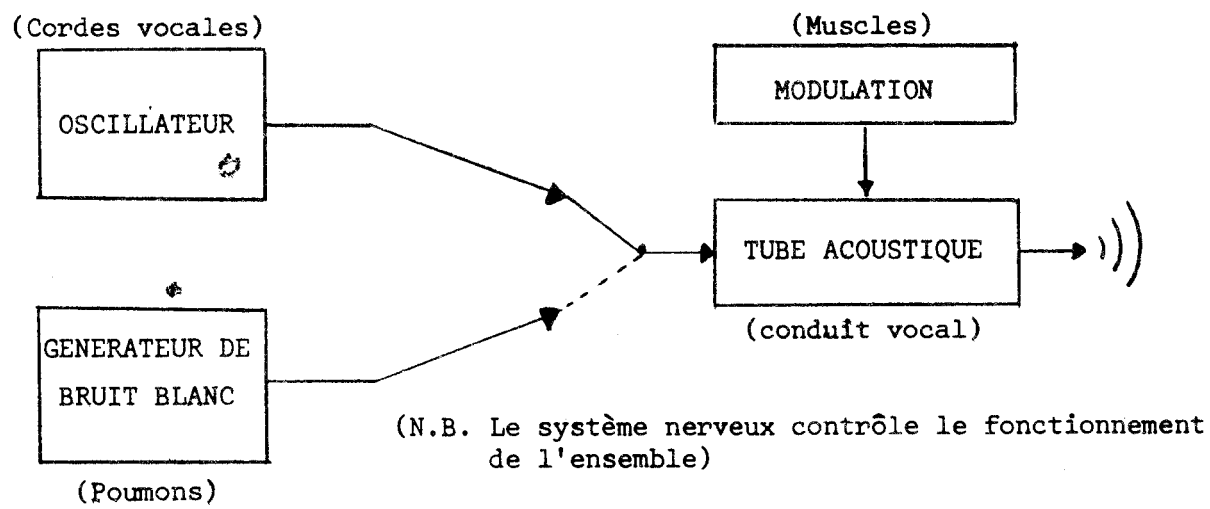
Cela confère au signal une bonne **immunité au bruit**, mais pose une difficulté relativement sérieuse au niveau du traitement :

- d'une part, il faut parvenir à une **compression** du signal permettant de réduire le volume d'information traitée (ou transmise)
- d'autre part, l'**extraction des paramètres pertinents** s'avère extrêmement délicate : il s'agit en effet d'extraire dans les plusieurs K-bits qui constituent en moyenne un phonème "acoustique", les 5 ou 6 bits qui permettent de coder le phonème "linguistique".

2) Continuité du discours : la phrase parlée ne peut se réduire à une simple juxtaposition des unités élémentaires (phonèmes, par exemple) du langage. En effet, la prononciation des phonèmes est modifiée par leurs voisins : il faut donc prendre en compte des **règles contextuelles**. D'autre part, sur le plan purement physique, les frontières entre phonèmes ou entre mots n'apparaissent pas explicitement. Il faut observer des indices permettant des présomptions de frontières : c'est le délicat problème de la **segmentation**.



a) Structure du signal vocal (linguistique)



b) Structure du signal vocal (acoustique)

Fig 22 : Un modèle simplifié de production du signal de parole

3) Variabilité du discours : Si, sur le plan linguistique, les phonèmes sont considérés comme des caractéristiques constitutives de la langue, il en va autrement sur le plan acoustique. En fait, on rejoint ici le problème du décodage dans la mesure où l'on ne peut pas de manière absolue définir des "invariants" phonétiques. Le signal souffre en effet d'une assez grande variabilité inter-locuteurs (fondamental, rythme, accent, ...). Dans l'idéal il faudrait donc discriminer dans le signal les aspects propres au locuteur, sur lesquels se fonde la reconnaissance du locuteur, et les aspects sémantiques constitutifs du message.

4) Multidimensionnalité : Nous entendons ici le fait que le signal vocal se réduise difficilement à une observation dans un espace unique, ou à une analyse dans une structure unique :

- sur le plan acoustique, en particulier, les approches reposant sur le temps ou (exclusivement) la fréquence s'avèrent incomplètes, même si elles donnent des résultats. Cela est dû à la double "personnalité" du signal vocal, tour à tour apériodique et stationnaire, mais globalement non-stationnaire. L'approche fréquentielle convient assez aux intervalles quasi-périodiques mais prend difficilement en compte les transitions rapides.

- sur le plan linguistique, la phrase du discours continu supporte, on l'a vu, une structure multiple, syntaxe, mots, (syllabes), phonèmes. A cela se superposent le plan sémantique (indispensable pour l'efficacité du système) et la prosodie (carctéristiques telles qu'intonation mélodie, accentuation, ...). Il est donc nécessaire du superposer autant de niveaux d'analyse.

Remarquons toutefois que si la multidimensionnalité acoustique, par les difficultés qu'elle introduit, joue un rôle négatif, la multiplicité de la structure linguistique par contre, joue un rôle plutôt positif : en effet, la coopération des différents niveaux d'analyse permet à ceux-ci de se compléter, voire de se corriger. Ainsi, par exemple, l'analyse au niveau lexical, par les prédictions qu'elle fournit, peut permettre de lever des ambiguités au niveau acoustico-phonétique.

IV-1.2. Paramétrisation du signal vocal

C'est l'étape essentielle du problème : en effet, il s'agit de pouvoir fournir au système des données issues du signal acoustique.

L'analyse du signal se fait en général (voir [113]) par des méthodes fréquentielles ou temporelles.

- l'analyse fréquentielle fournit un spectre du signal : celui-ci sera obtenu directement par échantillonnage des sorties d'un banc de **filtres passe-bande** ou d'un **vocodeur**. On peut également obtenir des résultats plus fins en effectuant des calculs de transformée de FOURIER. On travaille alors sur le **signal échantillonné en utilisant** la transformée de FOURIER discrète (DFT). Des algorithmes ont été proposés pour améliorer cette dernière : on les regroupe sous le terme **transformée de FOURIER rapide (FFT)**.

- l'analyse temporelle consiste à considérer un codage à court terme du signal par modélisation de l'appareil phonatoire. En pratique on se limite au conduit vocal et on montre qu'un filtre linéaire récursif convient assez bien. La méthode la plus répandue est le **codage par prédiction linéaire** (LPC) : on minimise l'erreur calculée entre l'échantillon obtenu par prédiction et l'échantillon réel.

Les méthodes précédentes ont un défaut assez gênant ; dans l'état actuel de la technologie, elles nécessitent une **puissance de calcul** trop importante — sauf peut-être dans le cas d'un câblage de cette fonction (vocodeur par exemple), mais dans ce cas les résultats fournis peuvent être insuffisamment précis—. De plus, on l'a remarqué (voir IV-1.1. 4)), étant relativement complémentaires quant aux phénomènes perçus, elles gagneraient à être utilisées concurremment.

Dans les cas où l'on souhaite fonctionner de manière moins performante mais en temps réel, ou pour compléter l'analyse classique,on recourt à des paramètres différents. Ceux-ci sont plus simples à obtenir mais certains d'entre eux relèvent d'un choix empirique. Cette "**paramétrisation simplifiée**" utilisera, par exemple :

- différence d'amplitude crête à crête
- nombre de passage par zéro (ZC)

- points de rebroussement
- fréquence de dépassement de certains seuils
- fréquence fondamentale (pitch)

Ce dernier indice, la **fréquence fondamentale**, est souvent employé séparément quelle que soit la méthode d'analyse. Il est en effet relativement important en tant que caractéristique : sa présence ou son absence donne une présomption sur le caractère vocalique ou non du signal. En outre, ses variations dans le temps participent à la détection des aspects prosodiques (intonation, mélodie, rythme, accentuation, ...).

En fait, cette étape de paramétrisation constitue le point critique du problème : les paramètres utilisés habituellement ne rendent pas compte de façon fiable des distinctions phonétiques, ce qui empêche le système de reconnaissance de n'être qu'un simple décodeur. Tant qu'une paramétrisation sûre du signal phonétique reste problématique, il faut pouvoir recourir de façon abondante aux contraintes de niveau supérieur (lexique, syntaxe, sémantique, ...) afin de réduire au minimum le contexte à envisager, c'est-à-dire les ambiguités possibles.

IV-1.3. Reconnaissance automatique de la parole (R.A.P.)

Il s'agit d'un domaine assez vaste dès lors qu'on considère l'étendue des approches possibles (dont témoigne le nombre de travaux qui lui sont consacrés). On peut toutefois essayer de caractériser de plusieurs points de vue les systèmes conçus ou proposés :

1) Type d'approche : On peut classer en deux grandes catégories les méthodes de reconnaissance de la parole :

- approche globale : on considère comme une "forme" indivisible les paramètres obtenus (par analyse spectrale par exemple). La reconnaissance consiste à confronter le signal à identifier à des formes de référence obtenues par apprentissage.
- approche analytique : on cherche à repérer et identifier les éléments constitutifs de la structure du signal (phonèmes, mots, ...).

2) Type d'élocution : les problèmes posés au système seront plus ou moins compliqués selon le type d'élocution qu'il tolère : ainsi, on distinguera des systèmes traitant des **mots isolés** (seuls ou séquences de mots séparés

par des pauses artificielles) ou des systèmes traitant la **parole continue** (proche de l'élocution naturelle).

3) Objectifs et contraintes : un certain nombre de caractéristiques, outre la précédente qui est primordiale, permettent de définir voire de restreindre le problème envisagé. Ce seront, en particulier :

- la **taille du vocabulaire**, la **syntaxe**, le **contexte sémantique et pragmatique** qui permettent de limiter les possibilités à envisager lors de l'analyse à ces différents niveaux. Réciproquement ces contraintes limitent le champ d'application du système.

- l'**indépendance vis-à-vis du locuteur** :le système envisagé pourra n'être que monolocuteur, ou au contraire être multi-locuteurs, c'est-à-dire indépendant des problèmes posés par un changement de locuteur.

- la **possibilité d'un apprentissage** : en fait, dans l'état actuel des recherches cet apprentissage s'avère indispensable. Il est évidemment fastidieux dans le cas de l'approche globale et l'on pourrait éventuellement le supprimer dans le cas de l'approche analytique, si les "invariants" phonétiques étaient parfaitement connus.

- l'**environnement physique** : ici, de nombreux points peuvent être soulevés ; citons notamment la sensibilité au **bruit** ambiant, l'introduction d'une **transmission** (cas d'un raccordement par téléphone) les **conditions d'acquisition** du signal (qualités du micro, distance micro-lèvres, etc ...).

Nous évoquerons rapidement le domaine de la **reconnaissance de mots isolés**. Le plus souvent, une approche globale est suffisante. De ce fait, en raison des performances directement liées à la taille du vocabulaire (temps de réponse et espace mémoire) celui-ci sera nécessairement réduit. La plupart des systèmes commercialisés (on trouvera dans [117] un récent panorama de ceux-ci) sont de ce type. Les difficultés, bien que fortement atténuées par les restrictions acceptées au plan des objectifs, subsistent sur deux points :

- durée de prononciation : l'acquisition des formes étant basée sur un échantillonnage périodique des paramètres, les variations dans le temps de la durée des signaux affectent leur description — ce problème se pose d'ailleurs de même dans les systèmes plus ambitieux —. On doit donc

recourir à des méthodes de comparaison dynamique (cf. III-2.3.5.). Toutefois, il est possible que le procédé d'acquisition soit relativement insensible aux erreurs ainsi introduites : notamment, si le vocabulaire est très réduit et le système monolocuteur, la paramétrisation permettant de distinguer les mots peut être très sommaire.

- variabilité inter-locuteurs : comme les problèmes de distinction inter-locuteurs sont encore mal résolus, deux façons assez peu élégantes peuvent permettre d'y faire face :

. limiter le système à être monolocuteur, c'est-à-dire que celui-ci fonctionnera de façon correcte pour le seul locuteur qui aura procédé à l'apprentissage.

. effectuer une pseudo-adaptation en enregistrant autant de vocabulaires d'apprentissage que de locuteurs acceptés, le plus simple étant alors d'identifier l'un de ceux-ci par un mot-clé (son propre nom par exemple).

Ces systèmes sont donc très contraignants au point de vue utilisation, et comme la taille mémoire exigée est proportionnelle au vocabulaire, leur marché s'avère relativement limité à des applications de bas de gamme.

En fait, on peut approximativement admettre que l'approche globale correspond aux mots isolés et que l'approche analytique correspond à la parole continue. En effet, la taille mémoire nécessaire pour stocker toutes les phrases productibles par une syntaxe (même réduite) et un vocabulaire donnés serait considérable. La reconnaissance de la parole continue passera donc nécessairement par une approche analytique. On peut toutefois envisager des exceptions à cette double relation entre le type d'élocution et le mode d'approche.

Ainsi, dans le cas de vocabulaires très étendus, une approche analytique de la reconnaissance de mots isolés peut être préférable : les données d'apprentissage sont en très petit nombre (phonèmes ou syllabes) et le vocabulaire prend beaucoup moins de place puisque chaque mot peut alors être transcrit sous forme phonétique ou syllabique. De plus, à supposer que l'identification phonétique devienne beaucoup plus fiable,

les systèmes deviendraient bien moins sensibles au changement de locuteur ou même aux variations intra-locuteur.

De même, l'approche globale, plus simple, peut convenir à deux aspects particuliers de la reconnaissance de la parole continue :

- la détection de mots : il peut être intéressant de détecter la présence de mots-clés dans la parole continue, soit pour simplifier l'analyse, soit pour déclencher une action particulière (démarrage/arrêt du système par exemple).
- l'adaptation au locuteur : un mot, plusieurs mots ou une phrase typique prononcés par chacun des locuteurs peuvent éventuellement permettre d'identifier un de ceux-ci.

Insistons enfin encore [113] sur la nuance importante entre **reconnaissance** et **compréhension** de la parole. En effet, dans ce dernier cas, on peut se contenter du contenu sémantique du signal sans l'avoir complètement analysé. Inversement il se peut que l'analyse terminée, une structure terminale étant reconnue, on constate que le message obtenu n'a aucune signification (en général, toutefois, l'analyseur sémantico-pragmatique sera chargé de détecter et rejeter ce genre de cas).

IV-1.4. Reconnaissance de la parole continue

L'objectif, à terme, est plus ambitieux : tenter de parvenir à un véritable dialogue homme-machine dans lequel l'élocution puisse être quasi-naturelle.

On a vu que deux sortes de motivations conduisent alors à recourir à une approche analytique :

- d'une part, l'impossibilité matérielle de recourir à la reconnaissance globale (taille mémoire, temps de recherche, ...).
- d'autre part, les informations multiples que produisent les différents niveaux de structuration du discours continu concourent à l'élaboration de la solution terminale.

En fait, l'ensemble de la tâche peut être considéré, théoriquement comme une **succession de niveaux d'analyse** (figure 23) faisant intervenir tour à tour les différentes sources d'information :

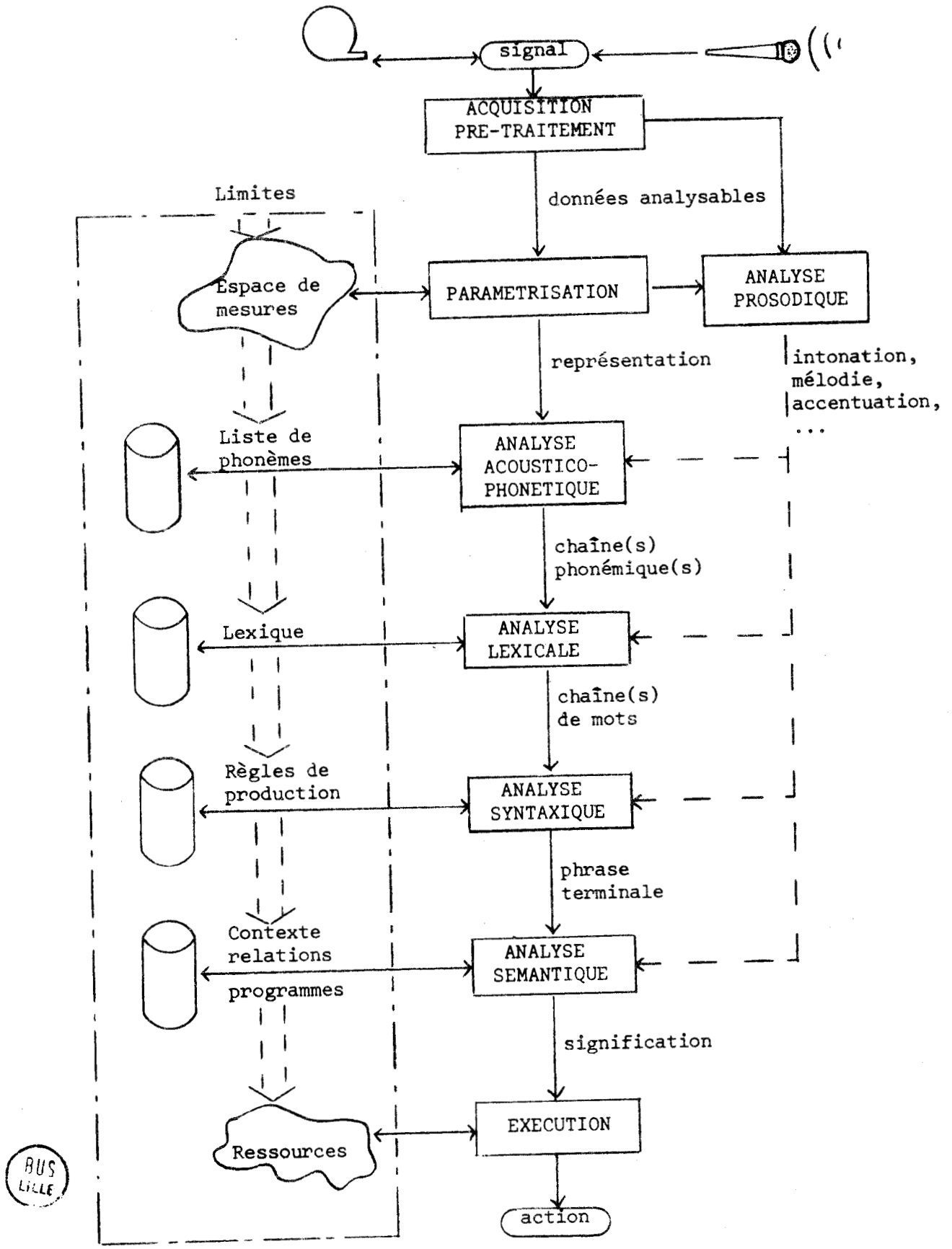


Fig. 23 : Décomposition logique de l'analyse de la parole continue

- acoustico-phonétique
- lexicale
- syntaxique
- sémantique
- pragmatique
- prosodique.

Ces sources d'informations peuvent être assimilées à des familles de processus dont l'espace de travail est constitué : de la structure de niveau inférieur qui leur sert de données, et de la structure qu'ils élaborent comme résultat. Chacun à son niveau aura à respecter des règles de construction. La prosodie joue un rôle particulier dans la mesure où les indices qu'elle fournit peuvent être pris en compte à différents niveaux selon leur étendue dans le signal : ainsi la vocalisation, l'accentuation, ou l'intonation, par exemple, interviendront respectivement aux niveaux acoustico-phonétique, lexical et sémantique.

Un système de reconnaissance de la parole continue doit impliquer la coopération de ces tâches. La conception d'un tel système est donc nécessairement complexe car les problèmes qu'elle soulève sont ardus et divers : en effet, on peut évoquer les plus importants :

- structures de données multi-usagers : par "usager" nous entendons ici une entité correspondant à chacun des niveaux de l'analyse. Chaque niveau, en effet, doit pouvoir accéder à la solution finale de l'analyse. Cela nécessite d'adopter une représentation des données standardisée qui permette à chaque tâche de prendre en compte ses données et de fournir ses résultats. Une variante consiste à ne tolérer des communications de données qu'entre tâches "voisines" dans la succession logique de l'analyse, correspondant à une organisation "pipe-line" du traitement.

- intelligence artificielle : chaque niveau d'analyse doit être doté d'algorithmes appropriés. Ceux-ci doivent tenir compte des possibilités d'erreurs dans les données qu'ils reçoivent, et, d'après les règles de construction, élaborer une solution (ou plusieurs). En outre, le système doit gérer la progression de l'ensemble vers la solution. Cela nécessite de lui faire suivre une stratégie de recherche

appropriée. L'intelligence artificielle intervient donc à l'intérieur de chaque niveau, d'une part, et, plus globalement, à l'échelle de l'ensemble de la stratégie de reconnaissance, d'autre part.

- synchronisation de processus : les tâches correspondant à chaque niveau d'analyse seront confiées à un processus, ou plus généralement à une famille de processus. Le système aura donc deux modes de fonctionnement : le traitement (parallèle ou séquentiel) des processus et la synchronisation des processus. A ce point de vue, les solutions, on le verra, peuvent être plus ou moins élaborées.

- architectures évoluées : certaines contraintes (notamment le temps réel ou la stratégie de synchronisation) peuvent déboucher sur la nécessité d'une architecture spécifique adaptée au problème. Deux aspects peuvent alors intervenir :
. introduction d'un certain degré de **parallélisme** permettant le multitraitement des tâches d'analyse.
. implémentation câblée ou microprogrammée d'une fonction particulièrement importante : ainsi peut-on envisager des **processeurs spécialisés** dans la FFT, le LPC, ...

Tous ces aspects reposent finalement sur la notion de coopération entre les différentes composantes de l'analyse. La **synchronisation** de ces tâches est donc un noeud essentiel du problème. Elle conditionne, en effet, tous les points évoqués ci-dessus, depuis la stratégie de conduite de la recherche jusqu'à l'implémentation. Elle influe directement sur les performances : en effet, un système dans lequel cette synchronisation consistera en un éveil séquentiel des différentes tâches sera nécessairement plus lent qu'un système permettant un fonctionnement simultané de ces tâches.

Sachant qu'une reconnaissance analytique de la parole continue doit nécessairement comporter tout ou partie des niveaux successifs déjà énumérés, on peut examiner quelles sont les différentes stratégies permettant de les faire coopérer.

IV-1.5. Méthodes de reconnaissance de la parole continue

Plusieurs publications décrivent et comparent les systèmes les plus importants réalisés ou proposés (notamment QUINTON [118], CARRE et al. [117]

On se réfèrera ici à la succession naturelle de l'analyse depuis les niveaux les plus bas (signal acoustique brut) jusqu'au niveau le plus élaboré (signification du message, voire action qui en résulte). On distinguera donc suivant le sens d'analyse suivi :

- des méthodes ascendantes, depuis le signal brut jusqu'à la réponse finale, en suivant en fait la décomposition logique du problème.

- des méthodes descendantes, en sens inverse, où des hypothèses générées aux niveaux les plus sûrs sont testées aux niveaux inférieurs.

- des méthodes mixtes, où les niveaux travaillent de façon asynchrone et concurrente à l'élaboration de la solution finale. On peut qualifier ces méthodes de "combinées" dans la mesure où toutes les tâches opèrent simultanément (au moins virtuellement) : seule la convergence vers le résultat final sert de critère.

IV-1.5.1. *Analyse ascendante*

C'est la technique la plus naturelle puisqu'elle suit la décomposition logique du problème (voir figure 23).

Chaque niveau d'analyse prend en compte les résultats fournis par le précédent. Il explore les constructions possibles d'après les règles de production (lexicales ou syntaxiques, par exemple) dont il dispose. Puis il fournit les hypothèses les plus satisfaisantes au niveau d'analyse suivant. Le plus généralement les résultats-hypothèses transmis sont au nombre de plusieurs, chacun étant affecté d'un score reflétant l'évaluation de la probabilité correspondante (figures 24 à 26).

Les avantages de cette approche sont les suivants :

- le résultat élaboré par le système est construit de façon "naturelle". De ce fait, la conception du système et de chacun de ses modules est plus simple.

- la communication entre les tâches est relativement simple ; en effet, chaque module reçoit ses données du même prédécesseur et les fournit au même successeur. La représentation des données doit donc simplement permettre un interfaçage entre deux modules particuliers. Ainsi, par exemple, le module lexical fournit un treillis de mots au module syntaxique.

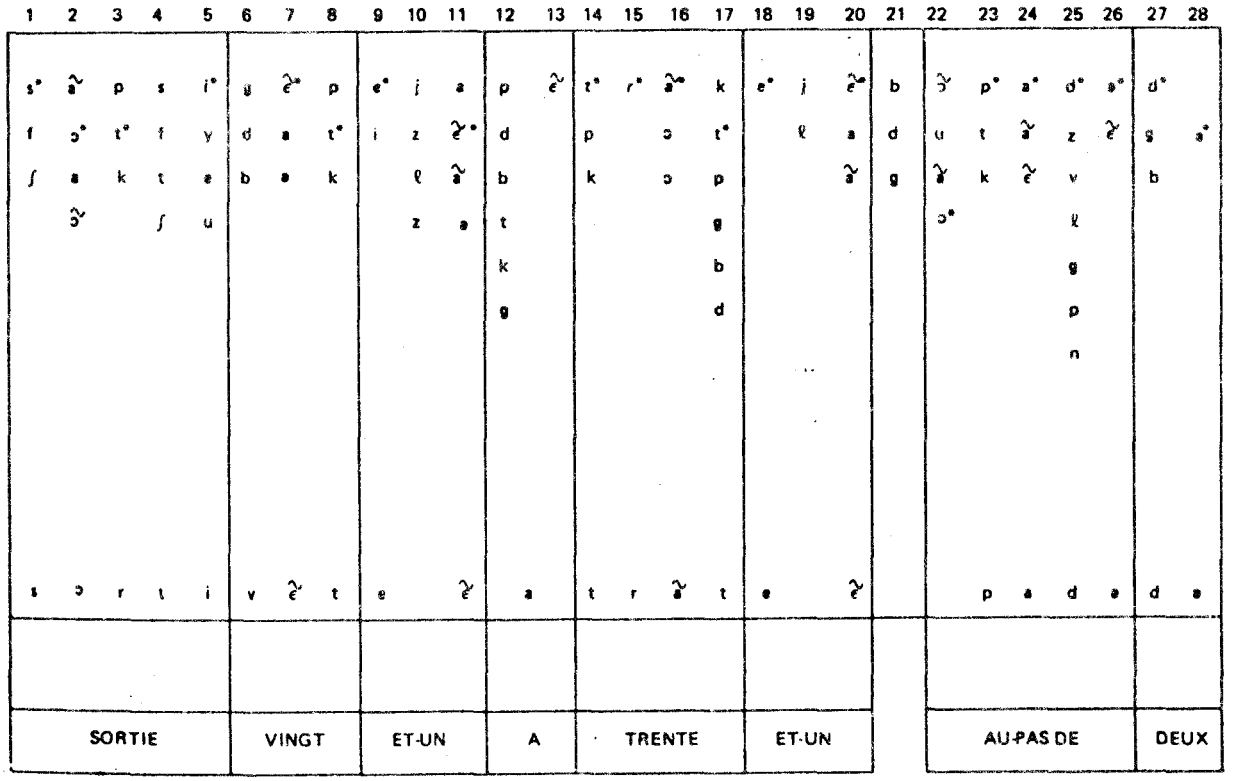
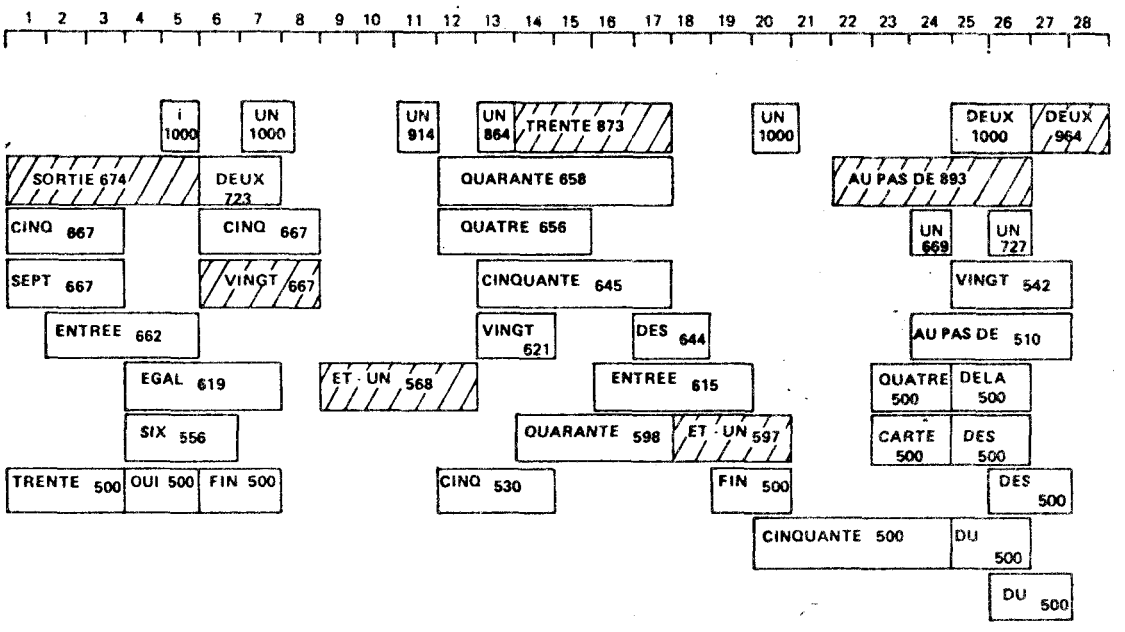


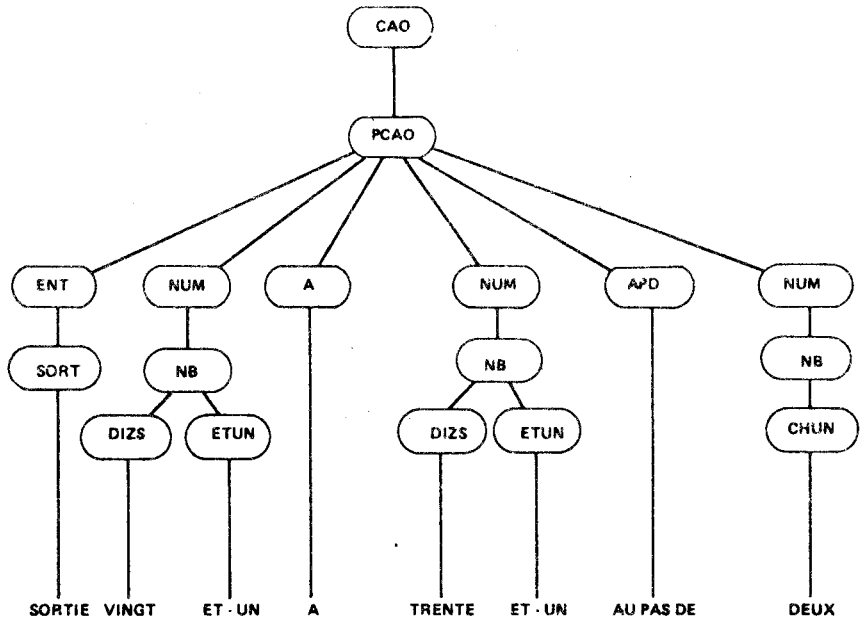
Fig. 24 : Résultats fournis par l'analyseur phonétique



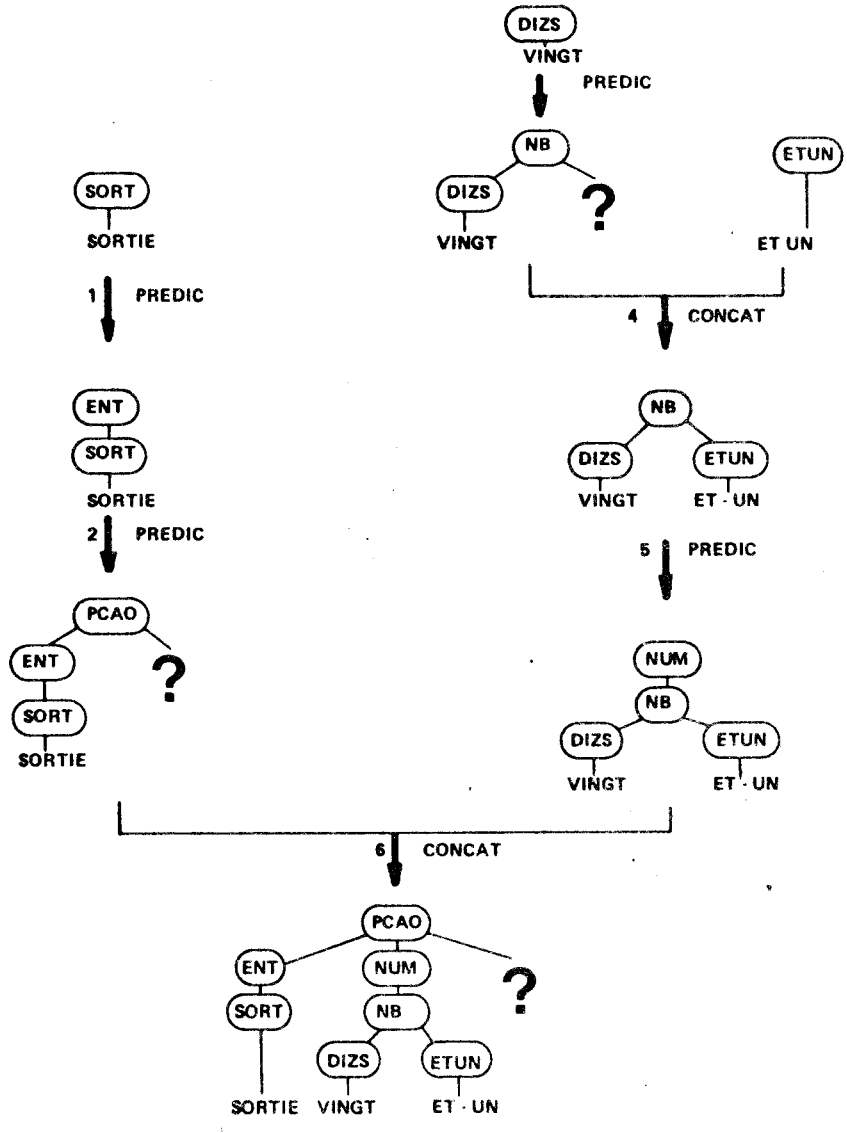
Résultats de la recherche lexicale sur la phrase : *Sortie vingt et un, à trente et un, au pas de deux.*
Chaque occurrence d'un mot est munie d'un score compris entre 0 et 1 000, qui indique la ressemblance entre le mot et le spectre phonétique.

Fig. 25 : Résultats fournis par l'analyseur lexical

(d'après le système KEAL [120])



a) Structure syntaxique de la phrase



b) Etapes de l'analyse syntaxique

Fig. 26 : Fonctionnement de l'analyseur syntaxique
(d'après KEAL [120])

- la **synchronisation** entre les tâches est extrêmement simple également : comme l'enchaînement des tâches est séquentiel et logique, celles-ci sont en fait synchronisées par les résultats. Remarquons qu'un tel système a pratiquement une structure "macro pipe-line" :le plus simple est donc un éveil des tâches par messages, chaque tâche éveillant son successeur en lui passant ses résultats.

Toutefois, un certain nombre d'inconvénients contrebalancent la simplicité de cette approche :

- le rôle critique des niveaux initiaux : en effet, l'ensemble d'un tel système repose sur les résultats fournis à l'entrée de la chaîne c'est-à-dire en particulier, les résultats de l'analyse phonétique. Or, on a observé qu'actuellement, cette étape est la moins fiable, compte-tenu des lacunes existant dans la définition des phonèmes.

- la croissance arborescente des hypothèses : corollairement à la remarque précédente, il faut conserver un nombre suffisant d'hypothèses de façon à s'assurer que la solution correcte apparaisse. En pratique, l'usage de seuils (déterminés empiriquement le plus souvent) permet, à chaque niveau, d'abandonner les hypothèses qui deviennent improbables.

- le temps de réponse : la conception même du système le voue à une exécution séquentielle des tâches sur un monoprocesseur : même sur une machine puissante, un temps proche du temps réel est alors difficile à obtenir.

Cette approche peut alors être améliorée pour pallier à certains défauts en lui offrant la possibilité de **recherche en arrière** (backtracking). Dans ce cas, on examine l'hypothèse la plus probable à un niveau donné en la poursuivant jusqu'aux niveaux suivants. Si le score obtenu devient trop faible, on remonte au niveau de départ pour prendre l'hypothèse de second rang, etc ... Une autre façon d'utiliser le backtracking est de garder constamment un nombre d'hypothèses égal (10 par exemple) et de remplacer au fur et à mesure qu'elles deviennent trop improbables les dernières classées.

Ce genre de système a suscité de nombreux travaux sur des machines classiques : il a au moins le mérite de permettre une mise au point et une amélioration progressives des différents modules du système. Parmi les exemples de réalisations, citons KEAL [118], [119], [120] au CNET, ou les recherches sur la parole continue chez IBM, abordées essentiellement sur le plan acoustico-phonétique [109], et basées sur la construction de graphes de transitions.

Notons enfin que cette approche, dans l'hypothèse d'une amélioration des modules phonétiques et du temps de réponse global (architecture adaptée ?), reste évidemment tout à fait valide.

IV-1.5.2. *Analyse descendante*

Compte-tenu des lacunes du niveau acoustico-phonétique, une approche en sens inverse peut s'avérer intéressante. On parle également de systèmes "guidés par la syntaxe et/ou la prosodie". (LEA [121]).

Le principe est donc de s'appuyer sur les informations "les plus sûres" pour générer des hypothèses aux niveaux syntaxiques et lexicaux puis les tester au niveau phonétique. Ces informations le plus souvent sont de 2 types :

. informations prosodiques donnant lieu à des hypothèses syntaxico-sémantiques limitées

. informations phonétiques de vraisemblance très élevée notamment certaines voyelles, les silences, etc ...

Cependant, bien que la génération d'hypothèses puisse guider le traitement des modules plus délicats — en fait, on travaille pratiquement en analyse-par-synthèse de façon prédictive —, cette méthode est plus complexe que la précédente à mettre en oeuvre.

En effet, les algorithmes d'hypothèse-et-test sont beaucoup plus difficiles à gérer que la recherche en faisceau pratiquée dans le cas précédent.

De plus, la nécessité de générer a priori des hypothèses de haut niveau impose le recours à un contexte syntaxico-sémantique beaucoup plus rigide et à un vocabulaire moins étendu. Autrement dit, une élocution quasi-naturelle pourrait entraîner un nombre d'hypothèses tout à fait considérable.

Des progrès dans le domaine de l'utilisation des indices prosodiques dans les niveaux supérieurs semblent nécessaires pour améliorer l'efficacité de ces méthodes.

IV-1.5.3. Analyse combinée

Plusieurs équipes se sont orientées, relativement récemment, dans une utilisation concurrente des deux techniques précédentes. Citons, par exemple, le système MYRTILLE II à Nancy [117] ainsi que certaines équipes américaines dont le meilleur exemple est celle de la C.M.U.

Dans ce dernier cas, il s'agit, en particulier dans le système HEARSAY-II [122], d'un **traitement en parallèle** des différents niveaux d'analyse et ce,de manière pratiquement indépendante et **asynchrone**. (C'est-à-dire avec un couplage inter-tâches relativement faible). On pourrait à la limite qualifier ce mode de traitement de "chaotique". Chaque niveau d'analyse participe de façon quasi-aléatoire à l'élaboration du résultat.

En pratique, le travail confié est finalement le même que dans les cas précédents : chaque processus, toutefois, génère ses hypothèses d'après l'état de la solution élaborée confrontée avec ses données propres (lexique, règles syntaxiques, ...). Les traits principaux de HEARSAY-II sont les suivants :

- un processus est associé à chaque niveau d'analyse. Chacun est capable de générer des hypothèses concernant le niveau dont il est responsable.
- les processus partagent une **base de données commune** ("blackboard") qui supporte les mises à jour progressives, par les processus, de la solution.
- les processus sont éveillés par l'apparition de **conditions** sur les données et non par la mise en sommeil d'un autre processus. Ces conditions servent en fait à communiquer les hypothèses.

- un **superviseur** gère la synchronisation des processus et contrôle la progression vers le résultat.

En fait, une implémentation efficace d'un tel système induit nécessairement le recours à une architecture multi-processeurs adaptée. Les caractéristiques de celle-ci doivent notamment permettre :

- la prise en charge des problèmes de synchronisation par évènements.
- le partage des données du "blackboard" entre des processus de nature différente.

HEARSAY-II a été implémenté sur Cm* [123] qui constitue une machine multiprocesseur très particulière. Malgré des résultats encourageants, il semble qu'on assiste à un certain recul où le "successeur" de HEARSAY, HARPY, s'oriente vers une implémentation sur mini-voire micro-calculateur. Actuellement, une telle évolution ne peut se faire qu'au prix d'une réduction importante des ambitions du système. Toutefois, de nombreuses applications justifient cette tendance.

IV-1.6. Problèmes essentiels et tendances actuelles

Il ne semble pas possible de trancher de façon nette entre les différentes stratégies évoquées. Les systèmes sont difficilement comparables tant au niveau des cahiers des charges (taille du lexique, langue, indépendance vis-à-vis du locuteur, etc ...) que des possibilités matérielles (coût du matériel, notamment).

Tout au plus, peut-on remarquer qu'un accroissement sensible des performances peut être espéré dans le recours à une structure multi-processeurs ; celle-ci peut alors être une **architecture parallèle** sur laquelle il semble qu'une stratégie "mixte" de type HEARSAY-II puisse fonctionner efficacement. A contrario, une **architecture pipe-line** dans laquelle chaque niveau d'analyse serait confié à un processeur pourrait être un outil efficace pour les méthodes ascendantes.

Au niveau de l'intérêt porté au domaine de la R.A.P., après un enthousiasme sans doute excessif, on assiste à un recul assez net. Cela coïncide avec la fin des projets ARPA-SUR* aux U.S.A. : ceux-ci ont en effet

* *Speech Understanding Research*

mis en évidence le manque de **connaissances fondamentales suffisantes** sur la parole, ce qui explique le manque de fiabilité des résultats des niveaux acoustico-phonétiques. D'autre part, contrairement à certaines prévisions le champ des **applications potentielles** s'est élargi.

De ce fait, les problèmes essentiels que pose actuellement la reconnaissance de la parole (quelle que soit la démarche utilisée) sont :
- l'amélioration de la **paramétrisation des phonèmes** de façon à tendre vers une définition des "traits pertinents" qui caractérisent les unités linguistiques qui composent le signal vocal.
- la définition précise des **objectifs** visés par les systèmes étudiés permettant de déboucher sur des applications réellement justifiées.
- l'étude de **moyens matériels adaptés** au problème facilitée par la diminution du coût relatif du matériel dans la conception des systèmes. En effet, si les machines classiques conviennent bien à l'étape expérimentale, il faut généralement, dans la plupart des applications opérationnelles envisageables, se rapprocher du temps réel.

Les travaux récents se situent donc dans deux perspectives différentes :
- à court terme : des applications très limitées sont étudiées et les systèmes réalisés sont opérationnels et même commercialisés [117]. Ces systèmes sont généralement des reconnaisseurs de mots monolocuteurs dont la technologie et le coût les vouent aux contextes mini et micro-ordinateurs. Toutefois, les techniques employées sont souvent simplifiées et ne concourent pas à la résolution du problème général. En outre, les sociétés qui lancent ces produits, bien que s'engageant certainement en s'appuyant sur un impact probable, vendent généralement leur matériel sans apparemment se soucier des services que celui-ci pourrait offrir : de sorte que la mise en oeuvre de ces systèmes semble plutôt du ressort de chaque acquéreur, individuellement.
- à long terme : des recherches plus fondamentales sont envisagées de façon à enrichir les aspects défectueux du problème. Les domaines les plus intéressants concernent :

. La **modélisation** du conduit vocal, de l'oreille, de la cochlée,etc... (CAELEN [124]) devrait permettre une meilleure approche de la production et de la perception de la parole. Notons qu'il s'agit de méthodes assez empiriques, mettant en jeu la physiologie, des expériences psychoacoustiques, etc ... Certains espèrent ainsi définir un modèle mathématique de **simulation**

de l'audition. Au minimum peut-on attendre de ces travaux une meilleure **définition des paramètres** à acquérir.

. l'**intelligence artificielle**, qui recouvre des domaines plus étendus, peut apporter des résultats précieux, en particulier pour la représentation des connaissances et les algorithmes d'analyse par déduction ou par induction, etc ... Cependant comme il s'agit de problèmes relevant de nombreuses applications (preuve de programmes, traitement d'images ...) cela déborde largement le cadre du traitement de la parole.

. la **conception d'outils matériels adaptés** : on a déjà évoqué le rôle que pouvaient jouer les architectures **multiprocesseurs** ainsi que la spécialisation vers certaines fonctions (paramétrisation, gestion des références, ...) de machines câblées. En effet, la plupart des applications envisageables dans le cadre d'un **dialogue** ne peuvent être satisfaisantes qu'avec un temps de réponse acceptable, proche du temps réel. De ce fait, la nécessité de réaliser un système spécifiquement approprié est impérieuse.

IV-2 - CONTRIBUTION DU TRAITEMENT ASSOCIATIF À LA R.A.P.

On a vu que le domaine du traitement automatique de la parole souffrait, entre autres lacunes, d'un manque d'**outils matériels** adaptés.

Parmi ceux-ci, on peut raisonnablement estimer que le **traitement associatif** puisse jouer un rôle positif. Outre le fait, déjà signalé, que les problèmes de reconnaissance de formes trouvent un terrain favorable dans l'**accès par le contenu**, le caractère **ensembliste et global** des manipulations associatives permet d'espérer un accroissement des performances des systèmes de R.A.P.

Il est intéressant, à la lumière des notions que nous venons d'évoquer sur le traitement de la parole, d'examiner de quelle manière on peut recourir aux mécanismes associatifs pour enrichir les systèmes de R.A.P.

IV-2.1. Extraction de sous-tâches associatives

Le chapitre I a clairement conclu sur le caractère partiel d'un système associatif ; en effet, les outils associatifs ne peuvent constituer à eux seuls la solution indépendante et autonome à un problème.

Il en résulte qu'il ne saurait se concevoir un système de R.A.P. utilisant exclusivement les mécanismes associatifs : ceux-ci interviendront donc dans la solution d'ensemble en tant qu'un (ou plusieurs) **sous-système(s)**. Autrement dit, comme les exemples évoqués le montrent, la solution au problème de la R.A.P. dans sa totalité, depuis l'acquisition des données acoustiques jusqu'à la réponse du système, ne peut que relever d'un **système global** mettant en oeuvre des aspects parfois très divers : selon les performances souhaitées et les circonstances de l'étude, l'implémentation matérielle reposera sur un "gros" monoprocesseur (comme dans KEAL par exemple) ou sur une machine très spécifique (comme Cm* dans le cas de HEARSAY II).

Les apports possibles du traitement associatif ne peuvent donc être envisagés que dans le contexte particulier d'un système d'analyse de la parole, et, par conséquent, dans le cadre d'un environnement matériel pré-défini.

Si l'on se réfère à la décomposition logique (cf. figure 23) de la démarche analytique de reconnaissance de la parole, il faut pouvoir y **identifier les tâches** qui relèvent de mécanismes associatifs.

Rappelons ici quels peuvent être les critères permettant de caractériser ces tâches :

- accès et manipulation des informations d'après leur contenu.
- opérations de type global sur un ensemble de données.

Toutefois, on a vu que certaines manipulations présentant ces caractéristiques pouvaient se ramener à un traitement séquentiel sur un processeur classique doté d'une mémoire adressable (voir chapitre III). Pour justifier un traitement séparé confié à un sous-système distinct, les sous-tâches extraites devront être suffisamment importantes et pénalisantes dans un système classique.

En fait, tout se passe comme si pour accroître les performances d'un système de R.A.P.,on devait extraire certaines tâches et les confier à une sous-machine spécifique. La "greffe" ainsi réalisée devra accroître notablement l'efficacité du système, le critère essentiel étant celui du temps de réponse.

Toutefois, le fait d'extraire de la solution d'ensemble certaines tâches,et donc pratiquement d'aboutir à une sorte de bi-processeur entre le système classique et le sous-système associatif,ne doit pas conduire à des communications coûteuses entre les deux parties. En effet, il serait inutile d'obtenir une diminution du temps d'exécution d'une tâche si le temps de transfert entre les deux parties du système vient compenser ce gain. Une troisième caractéristique essentielle vis-à-vis du reste du système est donc que ces tâches nécessitent un **faible volume de communications**, relativement au gain d'efficacité obtenu. Autrement dit, l'exécution de sous-tâches associatives par un sous-système adapté est conditionnée par un **couplage faible** entre ces tâches et l'ensemble du système d'analyse.

Enfin, notons une dernière contrainte qui comme la précédente relève du **type de méthode** de R.A.P. dans lequel s'insèrent les mécanismes

associatifs : si certains types de tâches sont identiques quelle que soit la méthode employée, d'autres peuvent varier au niveau des méthodes de traitement. En fait, à chaque niveau d'analyse, on confie toujours le même travail, les mêmes résultats sont produits, mais les méthodes, les algorithmes peuvent différer. Outre les exemples que nous détaillerons plus loin, citons par exemple l'analyse syntaxique où peuvent être utilisées différentes heuristiques notamment la recherche à retour arrière, le décodage séquentiel ou la recherche en faisceau (QUINTON [118]). Dès lors, selon la méthode utilisée, une tâche donnée correspondra plus au moins bien à des mécanismes associatifs.

IV-2.2. Aspects associatifs du traitement de la parole

IV-2.2.1. Données manipulées

Il convient tout d'abord de préciser quels sont les **objets manipulés.**

Dans l'approche analytique, on admet que la phrase prononcée ne constitue pas une forme complète, mais doit se décomposer comme la concaténation de mots (suivant une structure syntaxique) eux-même constitués de phonèmes.

Les objets ne forment donc pas une classe homogène mais changent **suivant le niveau d'analyse considéré.**

Les **objets de référence** dont on dispose peuvent donc être
. des phonèmes (décrits acoustiquement)
. des règles lexicales (permettant de construire le vocabulaire)
. des règles syntaxiques (propres au langage utilisé).
A ce propos, nous supposerons, pour simplifier, que dans la suite la reconnaissance analytique de la parole comporte trois parties :
- analyse acoustico-phonétique
- analyse lexicale
- analyse syntaxique (au sens large, sous-entendant la sémantique et le contexte pragmatique).

L'ensemble du système, du point de vue des données stockées, fera donc appel suivant le cas aux phonèmes, aux chaînes lexicales, aux règles syntaxiques.

Or, par rapport aux exemples de systèmes associatifs examinés en entamant cette étude, les objets stockés sont de nature différente : que ce soit dans le cas des bases de données ou celui du contrôle de trafic aérien, il était possible d'adopter une représentation relativement homogène. Par contre, ici, les informations manipulées par les différents niveaux d'analyse sont de nature très diverse.

Ainsi, les **phonèmes** sont-ils en général représentés comme des **séquences d'échantillons**. Ces échantillons peuvent être suivant le mode de paramétrisation adopté,des spectres fréquentiels, des vecteurs de coefficients prédicteurs (dans le cas de la prédiction linéaire), etc ... On peut donc les considérer comme des objets essentiellement **numériques**.

Le lexique est alors essentiellement constitué des successions possibles de phonèmes. De sorte qu'en assimilant chaque phonème à un "caractère", les informations traitées à partir de ce niveau sont plutôt **littérales**.

Une solution consisterait à représenter les objets de base, les phonèmes comme un vecteur d'attributs (cf. chapitre III). Ces **attributs** seraient obtenus soit grâce aux **paramètres** évoqués plus haut, (spectres fréquentiels ou vecteurs de coefficients de LPC) soit par des **propriétés**, extraites par calcul et caractéristiques de chaque phonème (voisé/non voisé, plosif, labial, ...).

Dans l'hypothèse d'une mémoire de référence unique, il faudrait représenter les structures lexicales par une méthode de type "fichier inverse" : cela conduirait à des chaînages entre phonèmes permettant de matérialiser leurs successions possibles, c'est-à-dire les mots valides. Cette solution s'avèrerait impraticable, en raison, d'une part, du nombre de chaînes à gérer (nombre de mots du vocabulaire), d'autre part des informations qu'il faudrait superposer à ces chaînes pour rendre compte des propriétés syntaxiques voire sémantiques (nature grammaticale, "type" sémantique, ...).

Il est donc plus raisonnable de stocker le vocabulaire comme un fichier constitué d'articles. Chaque article (représentant un mot) se

composerait des phonèmes correspondants, ou plutôt de leur code (leur "nom") auxquels s'ajoutent des informations évoquées précédemment, par exemple : "nom commun" ou "verbe transitif" etc ..., ainsi qu'éventuellement la représentation de l'orthographe du mot.

En fait, tout se passe alors comme si, aux différentes tâches d'analyse, on faisait correspondre un **sous-ensemble distinct** de la mémoire de référence. On peut alors considérer que les différents objets, phonèmes, mots, etc... appartiennent à des sous-ensembles distincts de la mémoire associative. A la limite, cela peut aller jusqu'à une séparation physique, notamment dans le cas où les tâches d'analyse correspondantes seraient confiées à des sous-systèmes **différents.**

Du point de vue des données, on peut donc considérer qu'il existe une frontière très nette entre les données acoustico-phonétiques d'une part, les structures de niveau supérieur (lexical, syntaxique, etc ...) d'autre part.

IV-2,2,2. *Types de traitements*

On peut également caractériser le traitement de la parole du point de vue des manipulations associatives qu'il induit.

Globalement, la question posée à l'ensemble du système prenant en charge la reconnaissance de la parole est du type "quelle est la signification qu'on peut associer à la parole prononcée". Dans l'approche globale ce serait de cette façon, littéralement, que serait effectuée la reconnaissance. Dans le cas de l'approche analytique, la requête concernant une phrase se décompose en

- associer une structure syntaxique
- associer une chaîne lexicale
- associer une chaîne phonémique

} à la phrase prononcée

Mais compte-tenu de l'imprécision du paramétrage acoustique, il s'agit en fait d'une **requête de meilleure ressemblance** et non de requêtes d'identité d'attributs. La question posée au niveau de la phrase s'exprimerait donc : "quelle est la signification **la plus vraisemblable** de la phrase prononcée".

En examinant mieux ce point de vue, on peut admettre qu'à condition de séparer le problème du codage phonétique du reste de l'analyse, on peut limiter les effets du "bruit" au niveau acoustico-phonétique. Autrement dit, comme les difficultés théoriques se posent essentiellement à ce stade, on peut distinguer deux types de requêtes dans le traitement :

- des requêtes de **meilleure_ressemblance** au niveau acoustico-phonétique. Elles fourniront un treillis de phonèmes constituant les meilleurs candidats.
- des **requêtes_d'identité** : en parcourant ce treillis le niveau lexical cherchera un (ou plusieurs) mot(s). Puis l'analyseur syntaxique pourra détecter une structure respectant les règles qu'il possède.

Notons toutefois que cette solution hypothétique conviendrait mieux à une méthode d'analyse ascendante. Mais surtout, elle supposerait une efficacité relativement poussée de l'analyseur acoustico-phonétique, dans la mesure où il faudrait que le mot correct, par exemple soit présent par tous ses phonèmes dans le treillis phonétique.

Dans le cas contraire —et c'est plus probable : il arrive en effet qu'un phonème de la phrase prononcée soit absent du treillis fourni par l'analyseur phonétique— il faut admettre que l'analyseur lexical puisse lui aussi fonctionner avec une certaine marge d'erreur, c'est-à-dire lui faire également effectuer des requêtes de meilleure ressemblance.

Plus généralement, les associations entre mémoires de référence et phrase prononcée, pourront se caractériser par l'un des 2 modes suivants :

1) Recherche exhaustive : dans le cas où il faut examiner toutes les données de référence, à un niveau donné, de façon à trouver le ou les objets présentant la **meilleure ressemblance** avec l'objet inconnu.

Ainsi, par exemple, pour chaque segment phonétique, on tente de fournir (la ou) les meilleures "étiquettes" (noms de phonèmes) possibles. De même au niveau lexical, un phonème pouvant être erroné ou omis par l'analyseur acoustico-phonétique, on cherchera de quel mot du lexique on se rapproche le plus.

2) Vérification d'hypothèse : il s'agit le plus souvent des niveaux supérieurs (c'est-à-dire à partir de l'analyse lexicale). La question est

simplement de savoir si telle chaîne (lexicale, syntaxique, ...) proposée par un autre niveau d'analyse, existe ou non. Ainsi, par exemple, les mots détectés par l'analyseur lexical permettent de construire plusieurs hypothèses de phrases que l'analyseur syntaxique vérifiera. Inversement celui-ci peut prédire à tel endroit de la phrase un nom propre dont l'analyse acoustico-phonétique n'aurait détecté qu'une syllabe : l'analyseur lexical vérifierait alors si ce nom existe.

Il est évident qu'ici encore méthode d'analyse et algorithmes, voire aussi support matériel, seront autant de contraintes qui permettront de définir de quel mode relève chaque type de tâche.

L'aspect essentiel réside dans le fait que ces deux modes correspondent aux deux familles d'interrogations associatives définies au chapitre III. De sorte que l'implémentation de ces deux modes de traitement posera des problèmes tout à fait différents : la vérification d'hypothèse est beaucoup moins coûteuse en temps de réponse mais elle suppose que l'un des niveaux d'analyse soit suffisamment fiable pour émettre des hypothèses peu nombreuses. Actuellement ce rôle ne peut être dévolu qu'aux niveaux supérieurs, compte-tenu de la faiblesse relative de l'analyse acoustico-phonétique. Ce mode est donc impraticable dans le cas de l'analyse ascendante, puisque les hypothèses ne seront construites qu'en fin d'analyse.

Pour insister sur cette distinction importante, on pourrait schématiquement représenter les deux modes de traitement associatif de la façon suivante :

1) "Soit une forme inconnue X, trouver $Q_i(F)$

$$Q_i(F) = \{A \mid d(A, X) = \min_{B \in F} d(B, X)\}$$ "

(c'est-à-dire A "ressemble à" X)

2) "Quelle est la probabilité pour que X ressemble à A"

ou mieux : "est-ce que X égale A ?"

IV-2.3. Intégration à la démarche de R.A.P.

On a remarqué (IV-2.1.) que l'extraction de sous-tâches associatives était étroitement liée au type de méthode de reconnaissance sur lequel repose le système. On a vu que des deux grands types de traitements qui composent l'analyse de la parole :

- calculs, évaluations, ...
- confrontations aux données de référence

ce sont évidemment ces derniers qui relèvent de mécanismes associatifs.

La **confrontation aux données** de référence peut être effectuée selon l'un des deux modes que l'on vient de décrire, recherche exhaustive ou vérification d'hypothèse.

Ce mode sera en fait largement conditionné par le **type de démarche** d'analyse de la parole.

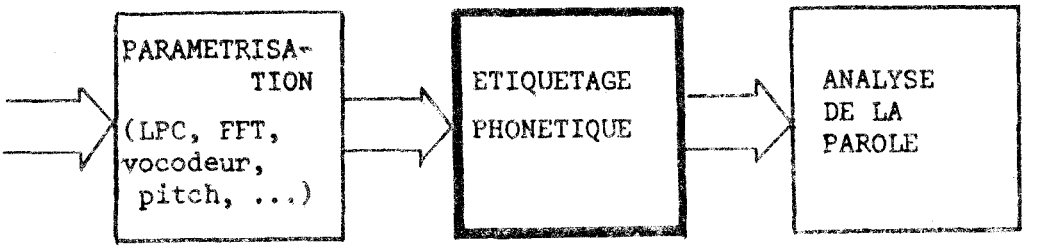
Pour illustrer ce point, il est intéressant de considérer le niveau le plus critique, l'analyse acoustico-phonétique. Celle-ci, en effet, nécessite, comme chaque niveau d'analyse, une comparaison avec les références dont dispose le système. Plus précisément, en général, le rôle de cet analyseur est d'affecter à chaque segment du signal inconnu une étiquette (ou plusieurs) représentant le (ou les) phonème(s) supposés reconnus.

Les résultats ainsi obtenus sont utilisés par le reste du système pour mener à bien les autres stades de l'analyse.

Cette phase est typiquement associative et l'on peut envisager de la confier à un processeur (au moins virtuel) spécifique. Le système de R.A.P. sera donc doté, dans cette hypothèse, d'un "processeur d'étiquetage phonétique".

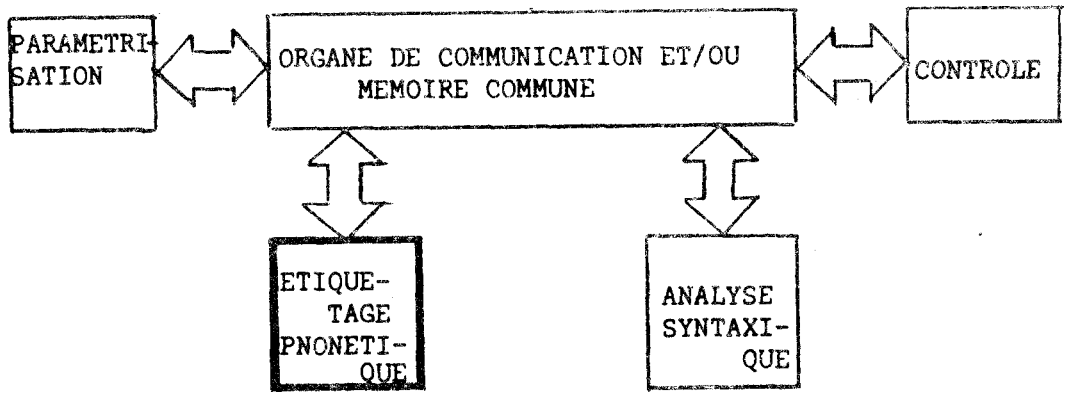
On peut alors considérer deux genres de contextes possibles :

1) Le système repose sur une méthode d'**analyse ascendante.** La structure du système peut donc être qualifiée de "macro-pipe-line" (voir ci-dessous) :



Le mécanisme associatif d'étiquetage peut être considéré comme faisant partie du pré-traitement. Il se synchronise sur le flux de paramètres caractérisant les segments (spectres, coefficients LPC, ...) et fournit une chaîne ou un treillis de phonèmes aux niveaux suivants de l'analyse. Comme aucune autre information ne lui est fournie que les paramètres calculés sur les échantillons du signal vocal, ce processeur d'étiquetage phonétique doit effectuer, au rythme des segments qui lui sont proposés, une recherche exhaustive dans les phonèmes de référence dont il dispose.

2) Le système repose sur une méthode d'**analyse descendante** ou, mieux, **combinée**. La structure du système est plutôt alors un multiprocesseur (par exemple : Cm* pour HEARSAY-II) :



Dans ce cas, la fonction d'étiquetage phonétique est assez différente : il s'agit de comparer avec les données de référence une (ou quelques) hypothèse(s) fournie(s) par l'un des niveaux d'analyse supérieurs. L'avantage au plan du temps de recherche est évident : l'espace mémoire qu'il est nécessaire d'explorer se trouve fortement réduit. Notons que c'est au prix, néanmoins, d'une complication de l'ensemble du système.

Cet exemple est assez significatif des relations étroites entre le mode d'exploration des données de référence, la méthode d'analyse, l'architecture du système, les performances. En outre, l'intervention de méthodes d'analyse phonétique plus fines jouerait certainement aussi un rôle.

En fait dans le premier cas, l'architecture du système est assez triviale, mais le processeur acoustico-phonétique doit être doté de possibilités relativement importantes, puisqu'il doit confronter chaque

segment avec l'ensemble des phonèmes de référence dont il dispose. Comme de plus l'évaluation de chaque phonème comme candidat possible n'est pas des plus simples, il est alors indispensable de conserver un choix de résultats suffisamment large pour ne pas pénaliser le reste de l'analyse.

Dans le second cas, au contraire, le travail est nettement moins fastidieux. Les hypothèses à évaluer seront, par exemple, du style : "comparer le segment [5, 7] aux phonèmes "a", "o", "œ". On tolèrera alors un processeur acoustico-phonétique moins performant mais on rejette la complexité vers le reste du système puisque celui-ci doit pouvoir générer des hypothèses suffisamment peu nombreuses pour ne pas retomber dans la difficulté précédente. Les indices permettant de construire ces hypothèses ne sont pas forcément faciles à détecter.

Plus vraisemblablement, compte-tenu des remarques faites dans les deux cas, il serait intéressant de faire coexister ces deux modes de fonctionnement, simultanément ou alternativement : un premier passage au niveau acoustico-phonétique se charge de repérer et d'identifier les segments phonétiques comportant les indices les plus "sûrs" - cela peut être simplement évalué par rapport à un seuil - Comme on ne traite que ces segments ("les plus faciles à reconnaître") et que leur identification est relativement aisée, on diminue sensiblement la puissance de traitement nécessaire. Ensuite, ces données phonétiques "sûres" peuvent être utilisées par les niveaux supérieurs de façon complémentaire avec les autres indices dont ils disposent (prosodiques, notamment). Les hypothèses ainsi construites seront alors plus précises. Aidée par ces hypothèses, l'identification d'autres segments phonétiques devient plus simple, et ainsi de suite ...

En fait, implicitement, c'est ainsi que fonctionne un véritable système "asynchrone" ; chaque niveau minimise sa tâche en effectuant a priori les tâches les plus faciles ; les résultats ainsi obtenus contribuent à réduire progressivement les ambiguités qui subsistent.

On a insisté, parmi les observations faites au chapitre I, sur les relations étroites existant entre les mécanismes associatifs et le

contexte d'application envisagé. Cela amenait à considérer ceux-ci comme un outil partiel contribuant à la solution générale.

On constate ici que l'approche utilisée pour résoudre l'ensemble du problème influe non seulement sur la possibilité de recourir aux mécanismes associatifs pour telle ou telle tâche mais surtout sur leur mode de fonctionnement lui-même, et par là sur leurs performances. La définition de ces tâches s'avère donc délicate puisque c'est principalement un accroissement de leurs performances qui peut justifier un recours au traitement associatif.

IV-2.4. Implications du mode de confrontation

De ce qui précède, il ressort clairement que seul le contexte particulier d'un système de R.A.P. bien défini peut conditionner les caractéristiques des mécanismes associatifs employés; notamment, pour chaque tâche qui y recourt, le mode_de_confrontation aux données de référence.

Toutefois, compte-tenu des différences importantes que cela induit sur les solutions envisageables, il est intéressant d'examiner succinctement sur des exemples les différentes approches correspondant aux deux cas.

IV-2.4.1. *Vérification d'hypothèse*

L'avantage de ce mode d'exploration est de nécessiter un domaine d'investigation assez restreint. Cette fonction de vérification d'hypothèse peut aisément reposer sur une **interrogation sélective** : Il sera alors possible généralement de faire appel à des techniques basées sur l'adressage : structuration des données, hash-code, etc ... Nous évoquerons deux cas de figure à titre d'exemples :

1) Recherche lexicale par stockage arborescent (Cette idée est décrite plus en détail dans [125]). Un lexique peut être aisément stocké dans une structure arborescente. Le contexte de la R.A.P. impose cependant une petite précision : les "caractères" constituant les mots du lexique sont en fait les noms de phonèmes. Cette structure présente un certain nombre d'avantages :

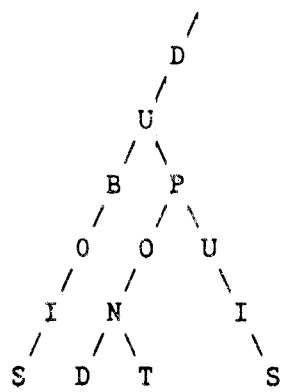
- elle est assez facile à représenter en mémoire adressable (techniques de liens) ainsi qu'à explorer.

- sa mise à jour est aisée (encore que peu utile en R.A.P.)

- elle convient particulièrement bien au stockage et à la recherche de chaînes.

Chaque objet, (ici, chaque mot), y est donc représenté par un chemin racine → feuille où chaque noeud correspond à une lettre ou plutôt un phonème. (Pour plus de clarté nous avons utilisé l'écriture alphabétique sur notre schéma à la place de l'écriture phonétique).

une "branche" d'arbre lexical (ici, on a affaire à des noms propres)



Un avantage supplémentaire réside dans l'économie de stockage obtenue pour les noeuds ou les branches communs à plusieurs chaînes. (Cet avantage est compensé, il est vrai, par la place occupée par les informations de chaînage).

Une telle structure est alors utilisable pour vérifier l'existence de n'importe quelle chaîne présentée comme hypothèse. Inversement, elle peut servir à construire une hypothèse pour le niveau inférieur à partir d'une hypothèse partielle qui lui a été fournie.

Il semble en outre intéressant, le cas échéant, de se servir d'un ordre (ici, il serait alphabétique) permettant à chaque niveau de l'arbre d'optimiser la recherche par dichotomie.

Exemples de fonctionnement :

1) Le treillis fourni par le niveau inférieur produit un certain nombre de chaînes hypothétiques. On peut alors vérifier que l'une d'elles existe bien.

2) Le niveau syntaxique a détecté un nom propre. On accèdera donc au sous-arbre qui leur correspond : cela fournira quelques hypothèses à vérifier par ailleurs.

2) Recherche phonétique par adressage : Toujours en supposant un fonctionnement par vérification d'hypothèse, il est possible d'envisager un stockage approprié des phonèmes. Pour simplifier, supposons ceux-ci, du moins les références "idéales" stockées, composés de 20 échantillons successifs, chaque échantillon comprenant 16 valeurs (coefficients de LPC ou niveaux d'énergie/canal ou encore paramètres plus "phonétiques" tels que vocalisation, explosion, ...); on suppose en outre un total de 40 phonèmes. On peut alors, par exemple, utiliser un indexage par rapport au nom des phonèmes en réservant une zone pour chacun d'entre eux. Ainsi une requête du style "comparer X et le phonème n° 23" se traduira immédiatement par une indexation en début de zone n° 23. Ensuite, l'organisation de chaque zone dépend de la façon dont doit être menée la comparaison avec le segment inconnu : on peut organiser le phonème par échantillons successifs, ou par valeurs. Mais, comme on l'a déjà remarqué, la vérification d'hypothèse ne soulève que peu de difficultés et cela laisse donc un assez large éventail de possibilités : on peut aussi travailler échantillon par échantillon (c'est-à-dire tous les échantillons n° 1 des phonèmes, puis les suivants, ...) ou valeur par valeur (par exemple pour suivre l'évolution d'un canal de vocodeur), etc ...

IV-2.4.2 Recherche exhaustive

Il s'agit bien ici d'une interrogation exhaustive de façon à satisfaire une requête de meilleure ressemblance.

Remarquons, dès l'abord, qu'il sera nécessaire, surtout dans le cas où plusieurs réponses sont exigées, de trier les résultats d'une manière ou d'une autre.

En fait pour illustrer l'importance du système environnant, on peut examiner sur un cas précis comment se déroule l'analyse acoustico-phonétique.

Dans KEAL [126] le module phonétique procède en plusieurs étapes :

1) Paramétrisation : un vocodeur à canaux fournit un vecteur de niveaux d'énergie.

2) On calcule un certain nombre de paramètres et l'on procède à un étiquetage sommaire des échantillons en voyelles/consonnes.

3) Segmentation en syllabes, sur la base de certaines indices caractéristiques.
4) Recherche de zones stables/instables dans les syllabes par localisation de la voyelle. Eventuellement cela modifie la segmentation et on repart en 3)
5) Classification grossière des segments phonétiques:voyelles, plosives, fricatives, etc ...
6) Identification des segments par fonctions linéaires dans $\mathbb{R}^n$. Les coefficients de ces fonctions sont obtenus par apprentissage.

Notons que ces étapes ne constituent qu'un exemple, mais l'analyse phonétique comporte toujours à peu près les mêmes mécanismes, parfois organisés différemment.

Ce qui est intéressant à souligner est que la majeure partie du traitement consiste à effectuer des calculs sur les données acquises.

De sorte que seule la phase d'identification des segments utilise une référence aux données acquise à l'apprentissage : dans ce cas, l'identification des phonèmes est finalement obtenue essentiellement par analyse des indices que l'on extrait du signal acoustique.

Dans un tel cas, la consultation de la mémoire est peu pénalisante : on propose à celle-ci une classe phonétique (par exemple, les voyelles) et la référence aux données stockées sert uniquement à affiner ce pronostic. Autrement dit, par exemple, pour un segment donné les étapes successives ont repéré ses frontières, l'ont classé comme consonne, puis comme plosive, puis comme plosive sourde. La consultation de la mémoire sert alors à tenter de savoir s'il s'agit de "p", "t" ou "k". On peut pratiquement considérer que la recherche associative est ici une vérification d'hypothèse.

Une autre approche phonétique consiste au contraire à travailler essentiellement sur les formes acoustiques et à les comparer de façon globale. Dans ce cas, on aura forcément à comparer le segment inconnu à toutes les formes de référence.

Ces deux voies diffèrent essentiellement par le rapport entre temps de calcul et temps de consultation de la mémoire. Dans le premier cas, les calculs d'extraction et de classification sont complexes mais la consultation de la mémoire est rapide. Au contraire, dans l'autre cas, il s'agit essentiellement de répéter une boucle de comparaison dynamique pour chaque forme de référence.

La **comparaison dynamique** que nous avons déjà évoquée, est rendue nécessaire par la possibilité de variations de la durée de prononciation. Une approche intermédiaire entre les 2 précédentes pourrait consister à représenter les formes par des valeurs en nombre fixe indépendantes de la durée et à extraire les valeurs correspondantes dans la forme inconnue. Il suffirait alors de comparer les formes valeur par valeur. Ces valeurs "orthogonales" au temps pourraient être des moyennes, des écarts d'amplitude, ...

IV-2.5. Discussion

Finalement, plus généralement, pour chaque niveau d'analyse, l'identification des séquences, qu'elles soient phonétiques, lexicales, syntaxiques, peut revêtir deux aspects :

- évaluation de paramètres, calculs de probabilités, c'est-à-dire en fait une démarche visant à effectuer la classification par examen direct successif de certaines propriétés dans la séquence inconnue.

- comparaison avec des séquences-types stockées en mémoire et étiquetage, affecté le cas échéant d'un score reflétant la probabilité estimée de l'étiquette.

Eventuellement ces séquences-types seront, dans le cas de l'analyse acoustico-phonétique, constituées de paramètres acoustiques (FFT, LPC, ...) ou d'un jeu de paramètres "pertinents" extraits de ceux-ci : dans ce dernier cas on utilise alors de façon combinée (dans une proportion

variable), la classification par évaluation de paramètres et la comparaison avec les références.

Du point de vue associatif, seule la seconde phase justifiera le recours à des mécanismes spécifiques. Outre la nécessité de consulter associativement les données de référence, il faudra vérifier les deux conditions suivantes :

- utilisation d'une **interrogation exhaustive**. En effet, dans le cas où la consultation se réduit à une simple vérification d'hypothèse, le plus souvent, un mécanisme d'adressage sera plus efficace.

- contraintes de **performances sévères**, et notamment un temps de réponse compatible avec le temps réel.

De plus, on a vu que les différentes méthodes de R.A.P. donnaient lieu à des modes de fonctionnement différents. Or, si la possibilité d'extraire du signal tous les traits "pertinents" permettant de caractériser les phonèmes s'avérait disponible et fiable, le système n'aurait plus besoin à ce niveau de recourir à des formes acquises par apprentissage. Toutefois, on peut considérer qu'une telle possibilité reste toute théorique, d'une part en raison des lacunes théoriques dans ce domaine, d'autre part compte-tenu du peu de souplesse que cela introduirait : en effet, même si l'on parvenait à définir assez précisément les phonèmes, ceux-ci subiront toujours des dégradations qu'il sera difficile de prévoir (bruit de transmission, rapport signal/bruit à l'acquisition, variations de durée, etc ...).

Une autre approche pourrait être envisagée dans le cas de l'**analyse combinée**. En effet, tous les niveaux d'analyse doivent évaluer, par rapport aux données dont ils disposent, la ou les hypothèses de solution qui s'ébauchent dans une mémoire commune. On peut alors imaginer chaque niveau d'analyse disposant de sa mémoire associative de référence. Sous contrôle d'un processus de synchronisation à définir, chaque niveau propose des hypothèses en comparant avec les références les séquences partiellement élaborées dans la mémoire commune. Ainsi, par exemple, le processus lexical pourrait-il trouver quelques mots candidats vérifiant à un endroit donné de la séquence inconnue la nature grammaticale proposée

par l'analyseur syntaxique et des phonèmes vocaliques. Le risque est alors d'aboutir à une croissance anarchique des solutions partielles : la difficulté est donc rejetée au niveau du processus superviseur. Toutefois les tâches d'analyse, elles,peuvent être simplifiées dans la mesure où l'on cherchera simplement les chaînes les plus vraisemblables, c'est-à-dire possédant le plus de "morceaux" communs avec la séquence inconnue. Pour reprendre les définitions au chapitre III, il s'agirait pour ces différents niveaux d'analyse de répondre à des requêtes d'identité partielle.

CHAPITRE V

PROPOSITION D'ARCHITECTURE ASSOCIATIVE

Dans ce qui précède on a pu mettre en évidence :

- la nécessité pour un système associatif de se situer dans un contexte d'application précis. Dans ce cadre, il faut alors définir les attributions du système associatif, car celui-ci ne peut concourir que partiellement à la résolution du problème posé.

- la difficulté à s'insérer dans une approche générale de R.A.P.* sans se référer aux contraintes imposées par celle-ci.

Pour proposer un "système associatif de traitement de la parole" on se trouve alors face au dilemme suivant :

- Si on colle de plus près au problème de la parole, il est impossible de concevoir le sous-système associatif sans l'intégrer à une démarche globale de R.A.P. Cela débouche à la limite sur la définition d'un système de R.A.P. ce qui dépasserait le cadre de cette étude.

- Si on tente de rester plus général, on ne peut plus parler de machine associative dans la mesure où celle-ci dépend fortement des données traitées et du type de manipulations souhaitées.

On choisit donc de présenter une proposition intermédiaire : les hypothèses posées sont suffisamment restrictives mais restent volontairement plus générales (en particulier, pour la terminologie introduite) que le strict contexte du traitement de la parole. Celui-ci sert alors, principalement au niveau phonétique, à illustrer la description mais constituerait cependant, avec les réserves émises plus haut, un domaine d'application privilégié.

Rappelons également qu'une réalisation matérielle est en cours. Elle présente un certain nombre d'aspects communs avec la proposition qui suit.

* Reconnaissance Automatique de la Parole.

V-1 - CONTEXTE ENVISAGÉ

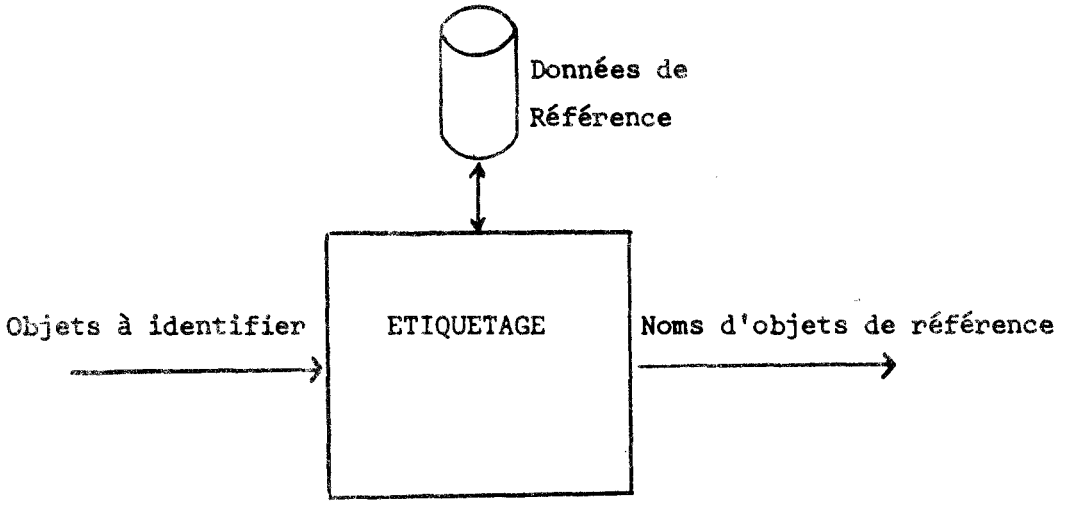
V-1.1. Définition du problème

Nous considèrerons que l'essentiel de la tâche envisagée consiste en l'**interprétation de messages "bruités"**, au sens large. Ce "message" se compose d'une séquence de symboles qui constituent un alphabet. Toutefois, on admettra que les objets que sont ces symboles sont à la fois peu nombreux et complexes à décrire. Cette complexité relèvera d'une part de la longueur du code nécessaire à leur représentation, d'autre part de la structure de ce code.

Autrement dit, il s'agira de reconnaître, dans un flux d'informations incidentes, les symboles, ou objets qui le composent. Cette approche est opposée aux tâches relevant des bases de données par exemple : en effet, dans ce cas, les objets manipulés sont nombreux mais de structure parfaitement définie.

On l'a vu (Chapitre IV), la reconnaissance de parole, par exemple, dépasse la simple identification des unités élémentaires du langage (phonèmes, syllabes, ...) : elle implique en effet des analyses de structures multiples (lexicale, syntaxique, ...) débouchant sur une action (commande de processus, accès à une information, ...). Toutefois, on a insisté sur l'originalité du niveau d'identification acoustico-phonétique, ce qui permet de le considérer de façon isolée (sans omettre cependant ses relations avec l'ensemble de la tâche de reconnaissance) : le problème est alors de choisir pour chaque segment de parole à reconnaître une étiquette phonétique (ou plusieurs, assorties d'un score).

Plus généralement, on considère donc des problèmes où il s'agit d'**associer à une suite de données inconnues des étiquettes** représentant chacune un des objets de référence.



Ces résultats sont éventuellement utilisés par le reste du système pour y subir d'autres analyses.

La caractéristique essentielle du type de tâche envisagé est que ces objets sont en nombre limité, mais que leur complexité (soit naturelle, soit due au processus d'acquisition) empêche la définition d'une fonction de décodage : on doit donc appliquer sur ces objets inconnus un traitement analogue à la reconnaissance de formes.

Outre la **reconnaissance des phonèmes**, d'autres types de problèmes peuvent s'apparenter à ces conditions. Citons, par exemple,

- l'analyse de scènes en robotique où il s'agit de reconnaître, parmi un certain nombre d'éventualités, une situation précise (position d'un objet géométrique, détection de pièces, ...)
- le traitement de données biomédicales, où l'ensemble des mesures physiologiques peut être interprété par rapport à un certain nombre de cas typiques (aide au diagnostic, ...)
- ou encore, la reconnaissance du locuteur, des caractères manuscrits, etc ...

Ces tâches relèvent en fait de **mécanismes associatifs** entre le flux de données inconnues et l'espace mémoire de référence (construit par apprentissage, ou par calcul). Compte-tenu de la taille de la représentation des objets (de façon à traduire leur complexité), il s'agira d'accès associatif à un niveau relativement global.

V-1.2. Données manipulées

Elles sont donc de 2 types :

- d'une part, un **flux séquentiel synchrone ou asynchrone d'informations incidentes** "inconnues".
- d'autre part, un espace de **données de référence** mémorisées.

Nous appellerons :

. segment : une fraction de longueur variable du flux inconnu à identifier, i.e. à qui il faudra associer une (ou plusieurs) étiquette(s) choisie(s) d'après les données de référence.

. modèles : les objets de référence stockés dans la mémoire et auxquels on confrontera les segments à identifier.

On suppose en outre que segments et modèles sont constitués d'une **séquence ordonnée** d'éléments de base que nous appelons **échantillons**. Ces derniers correspondent au mode de paramétrisation du phénomène observé.

Le caractère séquentiel et ordonné des segments sera le plus souvent imposé par le processus d'acquisition : dans la plupart des cas, les échantillons correspondront à des temps d'échantillonnage successifs du phénomène, et le segment, destiné à être confronté aux modèles pour identification, se composera des échantillons successifs contenus dans un intervalle temporel. Nous reviendrons sur les difficultés posées par la segmentation et les variations éventuelles dans le temps entre segments et modèles.

Nous noterons :

. $(x_o \ldots x_i \ldots)$ le flux d'échantillons inconnus. En fait, on supposera que ce flux est segmenté en $(X_1 \ldots X_i \ldots)$ où les X_i représentent les segments à identifier, avec $X_i = \{x_{io} \ldots x_{in}\}$

. $M_j (0 \leq j \leq p-1)$ les p modèles, en supposant qu'ils soient normalisés en longueur (nombre d'échantillons) d'où $M_j = (m_{jo} \ldots m_{jN-1})$ où N est le nombre d'échantillons par modèle.

V-1.3. Evaluation de la ressemblance

Le fonctionnement du système est assimilable à celui d'un automate associant à chaque segment du flux incident le nom d'un symbole de l'alphabet (phonème, pièce géométrique, caractère manuscrit, probabilité de diagnostic,...) Dans le cas où l'on souhaite conserver plusieurs solutions (les meilleures), il convient de prévoir un score reflétant la probabilité de détection estimée.

La particularité des tâches envisagées réside dans la complexité de l'évaluation (segment, modèle) qui conduit à cette association.

Une méthode pourrait consister en l'extraction de "paramètres pertinents" des segments inconnus. Cette approche aurait en outre l'avantage d'être insensible, grâce à un choix approprié de paramètres, aux dilatations ou contractions éventuelles des segments par rapports à leurs modèles. Dans ce cas, les modèles peuvent être simplement représentés par l'ensemble de ces paramètres. L'identification consiste alors à calculer les paramètres pour les segments puis à appliquer un algorithme de classification.

Ce type de tâche ne nécessite pas l'étude d'un système spécifique, et en particulier recourt assez peu à des techniques associatives d'accès aux informations.

En outre, cette solution repose essentiellement sur les propriétés des données traitées et suppose donc une connaissance approfondie de leurs caractéristiques. Cela entraîne dans certains cas le recours à un temps de calcul élevé.

Nous supposerons donc ici qu'il est nécessaire de travailler globalement, par "reconnaissance de formes" sur une représentation des modèles telle qu'on l'a décrite, homogène avec les segments. Toute l'information nécessaire à la reconnaissance reste ainsi disponible. De plus, la solution obtenue peut être espérée plus générale : en effet, indépendamment des paramètres pertinents propres à chaque problème, on pourra évaluer de façon globale la ressemblance entre segments et modèles.

Ainsi, en R.A.P., on a vu (chapitre IV) que la reconnaissance des phonèmes pouvait procéder par calcul de propriétés acoustico-phonétiques.

Toutefois, on a remarqué que cette méthode voit ses résultats, bien que relativement satisfaisants, plafonner en raison du choix parfois empirique des paramètres calculés. En outre, on rejette aux niveaux ultérieurs de l'analyse la détection d'erreurs éventuelles, ce qui peut compliquer l'ensemble du travail d'analyse.

Nous admettons donc, par hypothèse, que la tâche d'évaluation de la ressemblance s'appuie dynamiquement (c'est-à-dire au moment du traitement de chaque segment) sur la **représentation complète et brute des modèles** en mémoire. Compte-tenu des problèmes liés à la longueur variable des segments, il semble que la solution typique à ce genre de tâche soit la **comparaison dynamique** (cf. III-2.3.5.). Toutefois nous préférons malgré tout pouvoir envisager d'autres possibilités : plus précisément, il faut prévoir l'éventualité où un contexte d'application conduirait à une méthode différente d'évaluation de la ressemblance. Nous considérons ainsi, dans la suite, que l'évaluation (segment, modèle), quelles que soient les opérations nécessaires, repose sur une **consultation exhaustive des modèles**, représentés sous leur forme brute de séquences d'échantillons. C'est dans ce cadre qu'un outil associatif trouve toute sa justification.

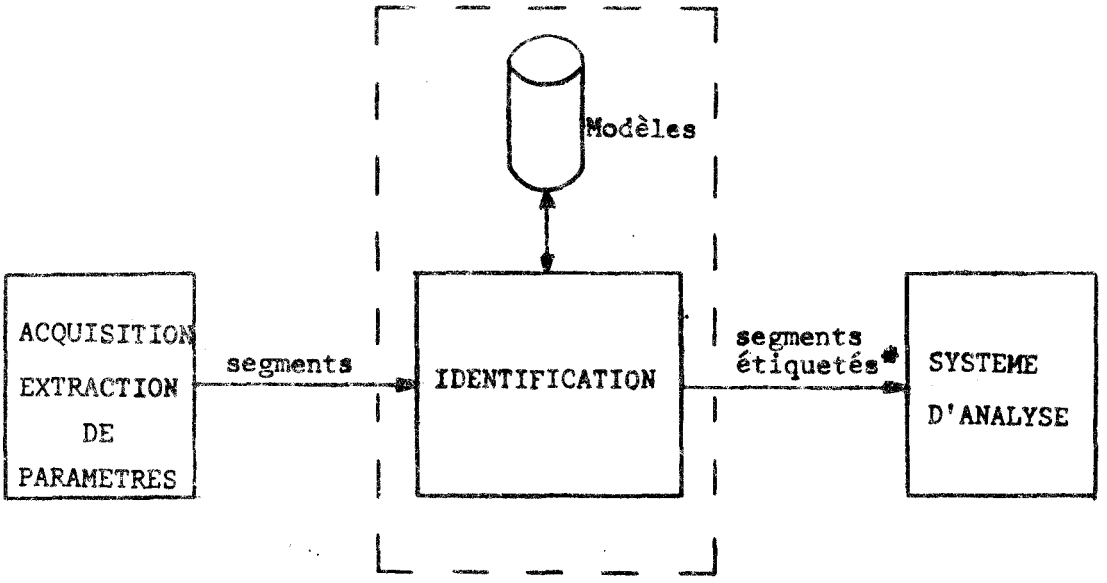
V-1.4. Insertion dans la méthode générale de reconnaissance

L'hypothèse dessinée jusqu'ici tient également compte de l'aspect partiel et incomplet de la tâche d'étiquetage. En effet, la plupart des applications évoquées recourent de façon plus générale à des algorithmes d'analyse relevant de l'intelligence artificielle. En fait, les mécanismes menant à la reconnaissance, voire à l'interprétation du message peuvent être classés en deux catégories :

- analyse (s) permettant de construire progressivement la (ou les) structure(s) du flux d'informations à "comprendre"
- identification des éléments de base, grâce aux données de référence.

Dans le cas d'une méthode d'**analyse ascendante** (cf. IV-1.5.1.), on commence par identifier les éléments "minimaux" du message (phonèmes, caractères manuscrits, ...). On a vu, dans l'exemple de l'analyse phonétique (IV-2.3.), qu'une des approches permettant d'isoler cette partie du problème consistait précisément à la rejeter vers l'amont du système.

On peut donc considérer que cette identification des symboles du message relève du **pré-traitement**. Dans la mesure où un sous-système spécifique peut se voir confier cette tâche, on peut espérer améliorer le temps de réponse global. La tâche d'analyse porte alors non plus sur les données brutes, mais sur les résultats fournis par ce sous-système "tampon" : ceux-ci consisteront en un (ou plusieurs) couple(s) (nom de modèle, score) associé(s) à chaque segment.



* *Informations du type* $(X_i, (M_j, S_{ij})..(M_k, S_{ik}))$ *où* M_j *est le nom d'un modèle et* S_{ij} *un score reflétant la ressemblance* (M_j, X_i).

Cependant, il peut être pénalisant d'exclure un "retour-arrière" sur ces résultats. En effet, les niveaux d'analyse peuvent constater des ambiguités sans pouvoir les résoudre :

On pourrait imaginer que le système d'identification puisse posséder deux modes de travail :
1) Evaluation (segment, modèle) rapide (en synchronisme avec le flux de données) pour tous les modèles, ce qui permet d'associer à chaque segment un (ou quelques) nom(s) de modèle(s) candidat(s).
2) Vérification d'hypothèse (segment, modèle) en utilisant alors un algorithme plus performant (mais plus long) permettant sur la base d'indices plus précis, d'affiner et de lever une éventuelle ambiguité.

Toutefois, tel qu'il est envisagé, le système de pré-traitement sera implémenté sur une machine spécifique, destinée à permettre une confrontation (segment, modèle) par consultation exhaustive des modèles mémorisés, au rythme d'acquisition des segments. Il s'agit donc essentiellement d'un traitement en temps réel, en synchronisation avec le flux de données. Il serait donc assez délicat de superposer à ce mode de fonctionnement un dialogue contrôlé par le reste du système. En outre, la vérification d'hypothèse implique des algorithmes différents, peut-être peu compatibles avec le mode de fonctionnement prévu pour ce système de pré-traitement.

On considèrera donc plutôt le travail de celui-ci comme une tâche fastidieuse mais approximative. Dans l'éventualité d'une vérification d'hypothèse, les besoins en données de références sont bien moins importants et un accès aléatoire suffit. L'algorithme de vérification, en outre, sera mis en ouvre de façon relativement rare. Il semble donc plus justifié d'inclure cette fonction dans le reste du système.

V-1.5. Caractéristiques déterminantes pour l'implémentation

Les aspects dont on a discuté jusqu'ici débouchent donc sur :

- la **mémorisation** permanente (à l'initialisation près), sous forme extensive, des modèles. Rappelons que ceux-ci, stockés sous une forme permettant de les confronter facilement aux segments, sont de nature complexe (au sens de: quantité de code/quantité d'information).
- l'évaluation pour tous les modèles de la **ressemblance** (segment, modèle)
- l'introduction d'un certain **parallélisme** permettant à cette tâche d'identification des segments de respecter les impératifs liés au temps de réponse.

En s'appuyant sur le reste de cette étude (en particulier chapitre I), l'introduction d'une **architecture associative** semble donc justifiée.

La puissance de traitement nécessaire à un fonctionnement compatible avec les contraintes imposées repose sur deux aspects : temps de réponse et taille mémoire.

Les paramètres qui influent alors sont :

- le nombre p de modèles gérés.
- la longueur N (nombre d'échantillons) d'un modèle.
- la fréquence d'échantillonnage, qui conditionne la durée d'un segment (en nombre d'échantillons variable) et le débit.
- le format d'un échantillon, qui conditionne la durée de l'opération de base.

On suppose que le nombre d'échantillons par segment n'est pas constant mais varie dans des limites "raisonnables" (i.e. peu en valeur relative à N).

Le temps_de_réponse repose donc essentiellement sur le débit des informations incidentes : la tâche d'évaluation des ressemblances (segment, modèle) pour tous les modèles doit durer, si possible, à peu près autant que l'acquisition d'un segment. Comme les évaluations de ressemblance entre séquences (voir III-2.3.5.) se décomposent à partir d'une métrique inter-échantillons, on peut estimer un temps de réponse par échantillon, ce qui permet de se référer à la fréquence d'échantillonnage.

La nature de l'algorithme d'évaluation est alors déterminante : le traitement implique un compromis calcul/accès mémoire qui influe sur les caractéristiques de l'architecture :

Supposons, par exemple, des objets représentés par 8 k-bits de code :

- l'évaluation peut consister en un calcul de la distance de HAMMING entre le segment et les modèles. L'opération élémentaire est simple (comparaison bit-à-bit) mais il faut accéder successivement aux 8 k-bits de chaque modèle.
- une comparaison dynamique peut être envisagée dans le cas d'une structure plus significative des modèles. Par exemple, si chaque modèle comporte 64 échantillons de 16 octets, la comparaison inter-échantillons devient plus complexe mais on ne doit plus accéder qu'à 64 "mots".

Le premier cas correspond à un traitement séquentiel par bits, le second introduit un certain parallélisme au prix de caractéristiques de format plus rigides.

La taille_mémoire, de son côté, doit prendre en compte, naturellement, la longueur de la représentation d'un modèle et est proportionnelle au nombre de modèles traités. Elle est donc directement déduite des paramètres cités plus haut. Toutefois, il convient de prévoir, selon la solution envisagée, les espaces de travail consommés par certains algorithmes ainsi qu'éventuellement, des zones permettant le stockage de code ou de microcode.

Les contraintes relatives aux performances, on le voit, sont très directement liées aux données manipulées. On vérifie ici le caractère spécifique d'une implémentation de fonction associative.

Dans le cas de la R.A.P., on peut, à titre indicatif, envisager une estimation des caractéristiques précédentes. Il s'agit uniquement de fixer un ordre de grandeur permettant ensuite de fonder un certain nombre de considérations sur la méthode de reconnaissance, l'architecture envisagée, les caractéristiques matérielles, ...

Les modèles doivent être des segments de signal vocal typiques. On sait que le français compte une trentaine de phonèmes. Nous admettrons qu'on puisse utiliser, de façon, plus générale, 64 modèles d'objets à identifier.

Compte-tenu de la redondance du signal vocal, des fréquences d'échantillonnage théoriquement insuffisantes permettent d'obtenir (l'expérimentation le montre) des résultats satisfaisants, au prix de défauts de détection de certains phénomènes rapides, caractéristiques de segments consonantiques.

On peut admettre que 100 Hz constitue une valeur acceptable, ce qui conduit à des segments de quelques dizaines d'échantillons, chaque échantillon comprenant environ 16 octets (ce qui correspondrait à 15 canaux de vocodeur et au pitch).

Ainsi l'espace de référence pourrait nécessiter environ, en majorant, 1 k-octet par modèle. Notons que cela reste hypothétique et l'on pourrait imaginer une fréquence plus élevée, par exemple 500 Hz, mais un

codage des niveaux moins fin. Dans ce cas, les segments seraient beaucoup plus brefs, et une voyelle, par exemple, pourrait s'étendre sur plusieurs segments successifs ce qui est compatible avec la stabilité relative des zones vocaliques. La comparaison (segment, modèle) devrait être plus rapide (de façon à suivre la cadence d'entrée) mais elle serait plus simple, d'une part grâce à la plus grande fréquence d'échantillonnage (consonnes mieux paramétrées) d'autre part en raison de la quantité de bits moins importante. Il faudrait toutefois envisager un nombre plus élevé de modèles possibles en raison des variations des segments vocaliques en fonction du contexte consonantique.

En fait, compte-tenu de la largeur du format de base (pratiquement l'échantillon de 128 bits) c'est plus l'organisation du stockage et le degré de répartition du traitement qui conditionne les performances : la capacité, relativement modeste, semble peu contraignante (surtout dans l'hypothèse d'un recours aux moyens séquentiels de stockage) :

Le **temps de réponse** semble plus exigeant : la durée d'un segment est de l'ordre de plusieurs dizaines de millisecondes. Il faut donc pouvoir assurer l'évaluation puis le choix du(des) meilleur(s) candidat(s) en un temps compatible avec cette durée. Nous reviendrons sur cette contrainte en estimant le nombre d'opérations nécessaires, notamment dans le cas de la comparaison dynamique.

Remarquons enfin, plus généralement, l'importance à ce niveau du compromis calcul/accès mémoire. Pour une évaluation sensiblement identique, un stockage des modèles sous forme beaucoup plus réduite (par calcul de valeurs indépendantes de la durée, par exemple) signifierait une quantité de calcul supérieure, en particulier sur les segments incidents. La différence essentielle tiendrait à la puissance matérielle nécessaire, les fonctions de traitement étant plus coûteuses que les fonctions de stockage.

V-1.6 Difficultés liées au nombre variable d'échantillons

Nous avons évoqué (III.2.3.5.) le problème posé par le traitement de requêtes associatives sur des objets de longueur variable. Pour le type de tâches que nous venons de décrire, il s'agit en fait de difficultés liées au **facteur temporel** dans l'acquisition des données. En particulier on a

signalé (IV-1.1.) l'influence de ce phénomène dans la reconnaissance automatique de la parole.

Pour tendre à rendre la tâche de reconnaissance insensible à ce genre de phénomène, plusieurs approches ont été évoquées (normalisation temporelle, extraction de paramètres moyens indépendants du temps, comparaison dynamique ...).

En fait, les difficultés de ce type peuvent se manifester ici de deux façons, que nous désignerons comme des problèmes de **cadrage** et de **facteur d'échelle** :

a) Problème du cadrage

Comme on a supposé qu'il fallait identifier chaque segment du flux incident par rapport aux modèles, cela impliquait, on l'a vu, de résoudre le problème de la segmentation.

En fait, le flux incident d'échantillons, généralement, apparaît continu sous la forme :

$$(x_o, x_1, x_2 \ldots x_i, \ldots)$$

Comme les modèles sont sous une forme fixe (éventuellement normalisée en longueur) il faut pouvoir faire correspondre un échantillon initial x_o aux m_{jo} de façon à pouvoir faire correspondre un des modèles M_j à un segment X_k contenu dans le flux inconnu.

b) Problème du facteur d'échelle

Les X_k sont de la forme $(x_{ko} \ldots x_{ki} \ldots s_{kn})$. Dans le cas général, n n'est pas constant et il s'avèrera que le modèle M_j tel que X_k R M_j (soit : X_k ressemble à M_j) compte un nombre d'échantillons différent de n.

Cette difficulté provient de l'**insertion** ou de l'**absence** d'échantillons dans le segment inconnu par rapport à sa forme "idéale" représentée par le modèle M_j. Dans le cas de la R.A.P., par exemple, ce sont les variations de durée d'élocution qui sont en cause.

Si les différentes solutions possibles — et nous prendrons comme représentant typique de celles-ci la comparaison dynamique — parviennent à "absorber" cette dernière difficulté, le problème de cadrage apparaît plus délicat. Deux types d'attitudes sont envisageables a priori :

1) On peut supposer le processus d'évaluation de la ressemblance conçu de façon à détecter dynamiquement un début de modèle. Il faudrait pratiquement, pour chaque modèle, démarrer une évaluation à partir de chaque échantillon et l'abandonner si l'on dépasse un certain seuil de ressemblance. Outre les difficultés à résoudre pour implémenter une telle solution (en particulier, détermination du seuil, et gestion d'un recouvrement élevé des tâches) cette approche s'avère relativement injustifiée : en effet, le type des données traitées et du mécanisme associatif (meilleure ressemblance) la rend caduque, en raison du caractère global, presque statistique de la ressemblance (segment, modèle). En fait, chaque échantillon, pris isolément, est dénué de sens, et c'est seulement a posteriori, sur un certain nombre d'échantillons qu'une ressemblance s'avère décidable : d'où **l'impossibilité de procéder progressivement, échantillon par échantillon.** Cela ne serait pas vrai dans le cas du décodage d'un message "déterministe" au niveau des symboles qui le composent (caractères ASCII, par exemple) où un simple automate à mémoire pourrait détecter des séquences données.

2) Le flux incident peut être constitué de façon telle que le problème de cadrage ne se pose pas. Cette approche est relativement hypothétique et rejette la difficulté en amont du système envisagé. Il faut en fait considérer que le flux se présente sous une forme segmentée du type :

$$X_o\ X_1\ \ldots\ X_k\ \ldots$$

où il s'agit simplement de mesurer la ressemblance entre chaque X_k et les M_j. Jusqu'à présent c'est ce qui a été implicitement admis. Cela crée donc un nouveau pré-traitement assez délicat, la **segmentation**. Il n'est évidemment pas question de se baser sur un découpage à partir d'un nombre moyen d'échantillons ; en effet, les conséquences conjuguées de l'erreur de cadrage et du facteur d'échelle pourraient mener à un glissement tel qu'aucune détection ne serait plus possible. Dans la plupart des cas, il faudra utiliser une **source d'information supplémentaire**, liée au type d'application, permettant de servir d'indice éprouvé pour la segmentation.

En fait, cette hypothèse d'une segmentation préalable est moins restrictive qu'il n'y paraît : en effet, un décalage revient à introduire des échantillons en début de segment : on peut considérer dans une certaine mesure qu'il s'agit d'un phénomène de dilatation (facteur d'échelle), pris en compte par la méthode d'évaluation insensible à la durée (comparaison dynamique). Cette observation n'est valide que si $\frac{\Delta n}{n}$ reste raisonnable, (ou si l'algorithme de comparaison dynamique est relativement souple).

On peut donc raisonnablement estimer que les erreurs dues à la segmentation seront absorbées par la comparaison dynamique, à condition que leur importance relative reste faible. Or, les contextes d'application envisagés permettent de considérer comme réaliste cette condition. En effet, dans certains cas, la segmentation donne des résultats absolument fiables, ou même est obtenue implicitement dans l'acquisition de l'information (objet séparés sur un tapis roulant, caractères manuscrits majuscules, etc ..). Dans les autres cas, une décision plus délicate doit être prise, même si les indices le permettant sont moins explicites : ainsi dans le cas de la R.A.P., au niveau acoustico-phonétique, la plupart des systèmes font précéder l'identification des phonèmes par un processus de segmentation basé sur l'énergie, les variations du fondamental, etc ... (Plus précisément, la segmentation en syllabes semble plus facile que la segmentation en phonèmes dans la mesure où les indices utilisés dans ce dernier cas sont obtenus après une pré-classification des phonèmes).

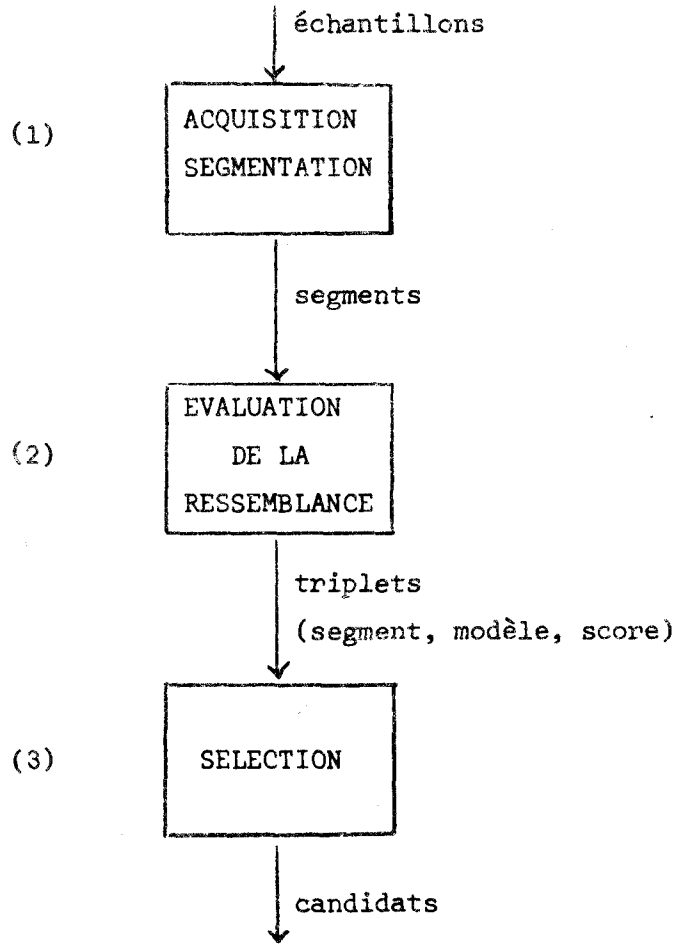
En résumé, en ce qui concerne l'influence du nombre d'échantillons, nous supposerons donc :

- la méthode d'évaluation insensible aux variations de durée entre segments et modèles
- le flux d'échantillons pré-traité par une méthode de segmentation
- les modèles normalisés en durée, dans la mesure du possible.

V-2 - ORGANISATION DU SYSTÈME

V-2.1. Décomposition des tâches

Si l'on considère l'ensemble du traitement à effectuer, depuis le flux d'informations brutes, jusqu'à la fourniture au reste du système d'une séquence de noms de modèles, on peut distinguer **trois phases** successives :



On appelle **candidats** les informations associées à chaque segment : elles consistent en un (ou quelques) nom(s) de modèle assorti(s) éventuellement d'une probabilité de ressemblance. Le cas échéant, on dotera la phase (3) de possibilités lui permettant, soit de réduire les informations transmises à un seul candidat par segment (mais cela peut s'avérer trop contraignant pour la suite du traitement), soit de transmettre simplement une liste (peu nombreuse) ordonnée de candidats.

La phase (1) consiste donc en une éventuelle mise en forme des informations et permet de fournir une suite de segments à identifier. Nous avons évoqué (V-1.6.) les problèmes posés par cette étape.

La phase (2) permet d'obtenir pour chaque couple (segment, modèle) une évaluation de la ressemblance exprimée sous la forme du score. C'est évidemment cette phase qui constitue la plus lourde tâche, ce qui justifie d'y porter un intérêt particulier.

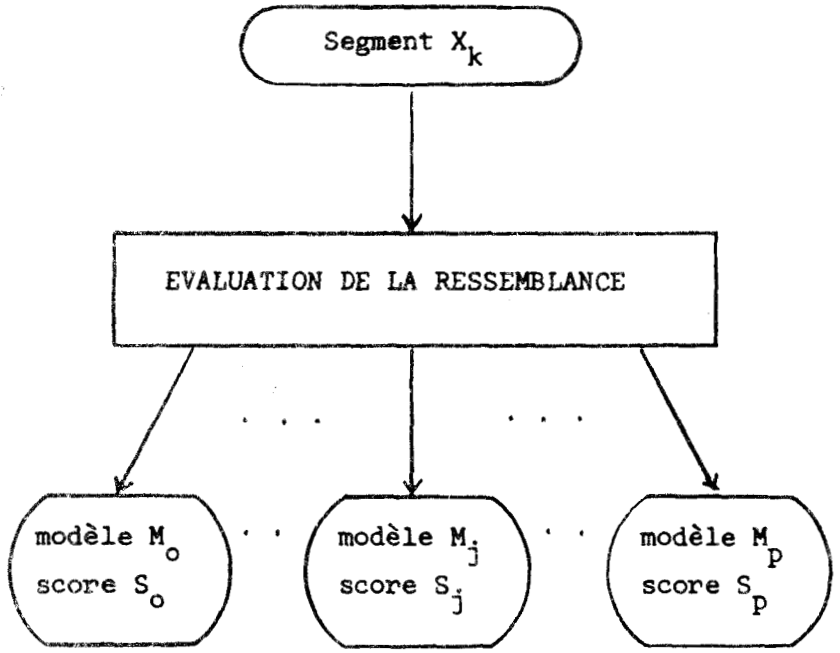
V-2.2. Expression du parallélisme

Si l'on examine plus particulièrement les données qui servent de communications entre les phases que l'on vient de décrire, on constate que le traitement est synchronisé par les données, et ce, en synchronisme avec le débit des informations acquises.

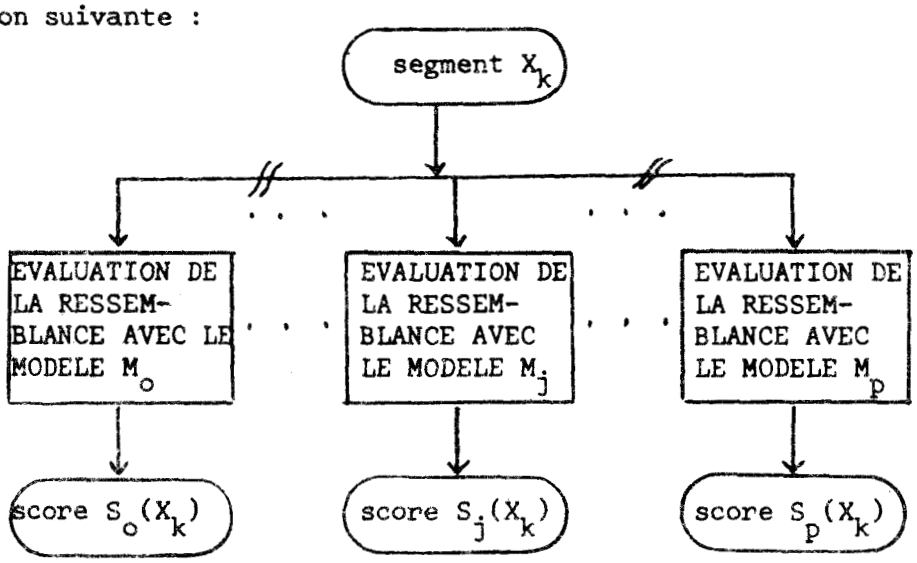
De cette manière, la décomposition séquentielle en trois phases distinctes fait apparaître une structure de type "macro-pipe-line", ce qui peut permettre, en traduisant cette structure au niveau de l'implémentation d'améliorer le taux performances/coût grâce à un certain recouvrement.

Toutefois, c'est la phase d'évaluation de la ressemblance qui, globalement, représente la charge la plus importante. La plupart du temps, dans un tel cas, on associe un processus à la forme inconnue (ici le segment). Ce processus accède (successivement) à l'ensemble des modèles pour évaluer les ressemblances (segment, modèle).

Or, la phase d'évaluation de la ressemblance, pour X_k, un segment donné, peut être schématisée de la façon suivante :



Du point de vue des modèles, en fait, il y a indépendance totale entre l'élaboration de chaque score S_j et les autres scores obtenus — à ceci près, éventuellement, qu'un modèle débouchant sur un score assez élevé réduit d'autant, théoriquement, la probabilité que d'autres modèles conduisent à des "bons" scores : cependant, cette information supplémentaire serait assez délicate à exploiter, à moins de simplifier fortement les possibilités du système en se satisfaisant d'un résultat dès qu'un certain seuil est dépassé (on rejoint ici les aspects signalés en III-3.1.)—.
La description du traitement peut alors être transformée et s'exprimer de la façon suivante :



On peut donc adopter une démarche opposée à la précédente et consistant à **associer un processus à chaque modèle**. De cette façon, le parallélisme inhérent à la tâche d'évaluation de la ressemblance se trouve explicitement exprimé. Le cas échéant, la prise en charge au niveau de l'architecture de tout ou partie de ce parallélisme doit permettre une amélioration sensible des performances dont la plus critique est évidemment le temps de réponse.

A chaque modèle M_j correspond donc un processus d'évaluation de ressemblance qu'on notera P.E.R. ; au besoin, on précisera P.E.R.j pour indiquer qu'il s'agit du processus gérant le modèle M_j.

Outre les motivations déjà exposées, un certain nombre d'aspects peuvent être envisagés à l'appui de cette démarche :

. Le type de problème considéré repose essentiellement sur l'espace de référence que constitue l'ensemble des modèles. Chaque modèle joue vis-à-vis du segment à identifier un rôle analogue, et donc, à la modularité des données permanentes, correspond la modularité du traitement.

. La correspondance processus/modèle permet de tenir compte, le cas échéant, d'une hétérogénéité des modèles en adaptant la tâche confiée à chaque processus en fonction du modèle qui lui est confié : comme les propriétés des modèles sont directement liées au contexte d'application, cette adaptation sera possible statiquement, à l'initialisation.

. Pour tenir compte de règles éventuelles de corrélation (voir III-1.1.2.) on peut conditionner l'éveil d'un P.E.R. d'après des résultats obtenus à l'occurrence précédente, ou moduler, d'après ceux-ci, l'algorithme d'évaluation. Dans le cas d'un processus d'évaluation unique, la mise en oeuvre d'un tel mécanisme serait plus problématique.

V-2.3. Dépendance inter-processus

Outre les P.E.R., qui constituent l'essentiel de la charge du système, deux autres processus, de nature différente vont jouer un rôle particulier.

L'un doit permettre de superviser le déroulement du traitement. Nous le désignerons comme étant le processus d'initialisation, P.I.

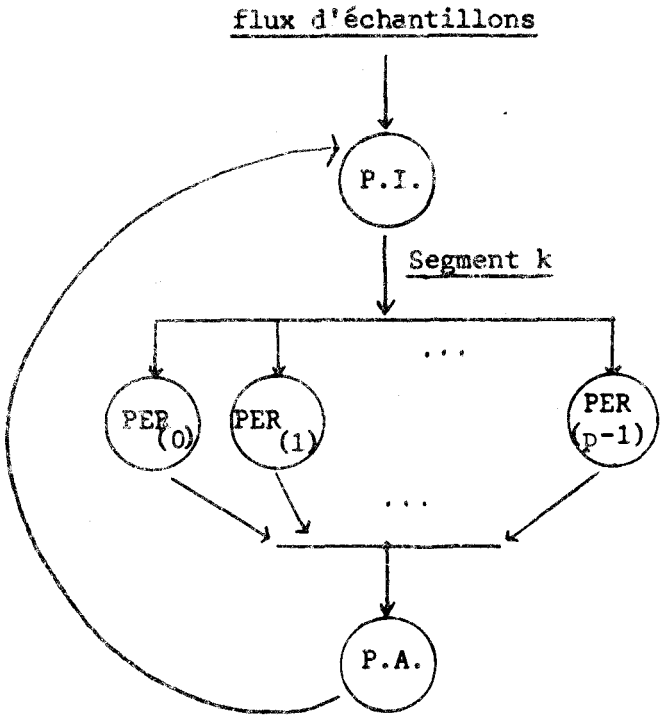
On pourrait admettre que le système ne comprenne pas un processus chargé explicitement de cette tâche : en effet, c'est essentiellement la réception d'un segment qui va conditionner l'éveil des P.E.R. : on pourrait donc considérer que le système soit contrôlé par les données. Cependant, il semble plus justifié que l'éveil des P.E.R. soit effectué par un autre processus : c'est en tout cas le mode de description que l'on adoptera. On peut alors admettre que le processus P.I. prenne en charge les tâches d'acquisition et/ou de segmentation.

Une contrainte impérieuse au niveau d'ensemble est de conserver, dans la chaîne de traitements, l'ordre séquentiel des segments. Cela impose donc que chaque occurrence de segment fournie par P.I. éveille les P.E.R. et que ceux-ci soient détruits avant l'occurrence suivante. Il est alors commode de représenter le fonctionnement de P.I. comme intermittent, son éveil étant conditionné par la destruction des P.E.R. correspondant à l'occurrence précédente.

Une autre fonction particulière est indispensable. Comme dans tout système associatif (cf. I-2.3. notamment), l'expression d'un certain parallélisme débouche sur la nécessité d'une fonction de synchronisation : en effet, la sélection d'informations parmi les réponses obtenues, ou l'élaboration d'une réponse globale à partir des réponses partielles imposent une tâche supplémentaire et différente de celle assumée par les P.E.R.. On appellera **processus d'arbitrage, P.A.**, le processus correspondant.

On pourrait imaginer qu'elle relève des P.E.R., c'est-à-dire que l'exécution des P.E.R. comporte une phase terminale de concertation mutuelle. Cette approche aurait pour effet d'alourdir sensiblement la complexité des P.E.R. : d'une part, la fonction de concertation suppose une puissance de calcul supérieure ; d'autre part, cela débouche sur des communications inter-P.E.R. ce qui réduit considérablement leur indépendance mutuelle (seul le segment peut être considéré comme dépendance indirecte, en tant qu'information partageable).

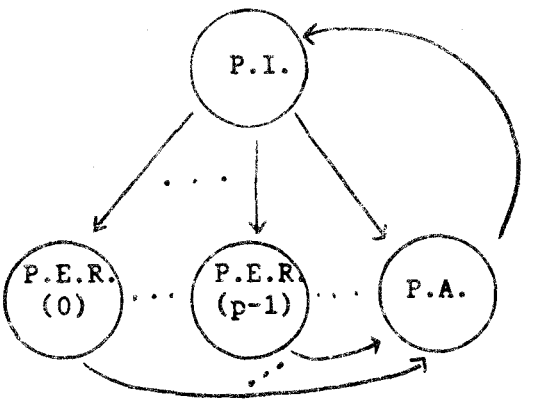
On peut alors représenter la dépendance entre les différents processus de la façon suivante :



Si l'on désire exprimer de façon plus nette le rôle joué par P.A., on peut considérer qu'il est créé par P.I. (qui lui fournit éventuellement des paramètres servant de critères de décision) : en effet, de même que pour les P.E.R., un P.A. doit être engendré à chaque occurrence de segment.

Notons une ambiguité dans la schématisation précédente. En effet, l'éveil de P.A. semble conditionné par la destruction (résultat obtenu ou échec) de l'ensemble des P.E.R.s. En réalité, P.A., comme les P.E.R., peut être éveillé par la transmission de la part de P.I. d'un segment : comme le contexte de l'interrogation exhaustive impose que l'évaluation aît eu lieu sur l'ensemble des modèles, c'est en fait la destruction de P.A. (consécutive à l'élaboration du résultat global) qui sera conditionnée par celle de tous les P.E.R.s.

A la limite, indépendamment des informations communiquées entre processus (en particulier résultats émis par les P.E.R.s vers P.A.) la dépendance des processus peut être décrite de la façon suivante :



Sous cette forme, apparaît plus clairement le parallélisme entre P.A. et les P.E.R.s du point de vue de P.I. En effet, pour un segment donné, P.A. a la possibilité de s'éveiller dès qu'**un seul** parmi les P.E.R. a abouti. De cette façon, les performances de l'implémentation peuvent être améliorées si l'on parvient à traduire cette relation en un **recouvrement** (même partiel) de P.A. et des P.E.R.s.

V-2.4. Processus d'évaluation de la ressemblance

On rappelle que chaque occurrence de segment donne lieu à la création d'un P.E.R. avec chaque modèle.

Les ressources matérielles indispensables à son exécution (processeur, mémoire de travail, ...) seront envisagées plus loin. Outre celles-ci, un P.E.R. doit disposer de plusieurs informations pour pouvoir être activé :

- la **représentation du modèle** dont il est gérant
- la description du traitement : c'est-à-dire la méthode d'évaluation (modèle, segment) à utiliser. On l'appellera, dans un sens large, **programme** (ce peut être, par exemple, une instruction décrite par microprogramme)
- le **segment** lui-même, qui a la particularité d'être partagé entre tous les P.E.R. qui lui correspondent.
- les **informations de contrôle** qui peuvent représenter d'éventuelles conditions prédéfinies liées par exemple aux résultats obtenus sur les segments précédents.

L'arrêt d'un P.E.R., i.e. sa destruction, est soumis à deux types d'évènements :

- obtention d'un résultat, ce qui permettra au P.E.R. de restituer un score. Il s'agit là de l'aboutissement positif de l'activité d'un P.E.R.

- survenue d'un échec, constaté soit de façon interne au P.E.R. (dépassement d'un seuil, par exemple), soit de façon externe, dans le cas où un mécanisme de surveillance ("chien de garde", par exemple) s'avère nécessaire.

Notons cependant au sujet de ce dernier point qu'une anomalie éventuelle conduisant à la destruction d'un P.E.R. par un mécanisme externe serait ici difficilement tolérable. En effet, ce genre de dispositif est généralement utilisé dans des contextes, tels que la mise au point de programmes en traitement par lots, où :

- d'une part, les causes d'anomalie sont notables (erreurs logicielles, divergence d'un algorithme, ...)

- d'autre part, l'effet de la destruction forcée est relativement peu dommageable (voire parfois assez positif).

Or, dans notre problème, le caractère global du traitement associatif d'un segment par rapport à l'ensemble des modèles rend critique, pour le résultat final, la sûreté d'exécution d'un seul P.E.R. (d'autant plus que l'on travaille en temps réel). On peut alors considérer que le caractère statique de la représentation des modèles et de la description du traitement leur impose, en fonctionnement normal, une certaine fiabilité. De plus les segments sont partagés entre tous les P.E.R. Une anomalie dans un P.E.R. ne peut donc être attribuée, en principe, qu'à des causes matérielles, ce qui ramène le problème au niveau de la tolérance aux pannes matérielles.

L'activité d'un P.E.R. peut donc être décrite en trois phases successives :

1) Création
2) Exécution
3) Destruction.

La phase de **création** correspond à une fonction de contrôle assurée par P.I. lorsqu'il fournit un nouveau segment.

La phase de **destruction** relève, elle, de P.A. à qui les P.E.R. transmettent leur résultat.

Les informations échangées dans ces deux phases de synchronisation seront détaillées plus loin.

En ce qui concerne l'exécution d'un P.E.R. on peut, par exemple, la schématiser de la façon suivante :

```
Début
Recevoir SEGMENT ;
Accéder à (MODELE, DIST) ;
DIST (SEGMENT, MODELE) ;
si ECHEC alors
        transmettre ECHEC
        sinon
        transmettre SCORE fsi ;
Fin
```

en adoptant les conventions suivantes :
- SEGMENT est la représentation du segment occurrent.
- MODELE est la représentation du modèle géré par le P.E.R.
- DIST est la description de l'évaluation de la ressemblance ; on peut ici l'assimiler à une procédure.
- SCORE est le résultat fourni par DIST, s'il existe
- ECHEC est un booléen, fourni par DIST signalant le cas où SCORE n'existe pas (dépassement d'un seuil par exemple).

On peut noter l'omission d'informations supplémentaires telles que les informations de contrôle ou même, par exemple, la valeur du seuil (transmise normalement comme paramètre à DIST).

En outre, la distinction, au niveau des informations reçues entre SEGMENT et (MODELE, DIST) est assez arbitraire : elle permet toutefois d'accentuer le rôle de SEGMENT dans la synchronisation en particulier pour la création du P.E.R. par P.I. Au contraire, l'accès à MODELE et DIST, qui sont des informations "privées" propres à chaque P.E.R. peut être considéré comme indépendant. Nous reviendrons plus loin sur ce point.

La séparation dans la description précédente des fonctions Accéder à ... et DIST (MODELE, SEGMENT) est elle aussi purement conventionnelle ; elle permet, en particulier, de mettre en évidence les deux

fonctions essentielles du P.E.R., **consultation** et **traitement**, c'est-à-dire en fait les deux types de ressource qui lui sont nécessaires. Cependant, une implémentation peut éventuellement recourir à un **recouvrement** de ces deux fonctions, spécialement en tenant compte du caractère séquentiel et ordonné des modèles ; il peut alors être envisagé qu'une décomposition appropriée du traitement puisse conduire à un accès et à un traitement échantillon par échantillon. Cette remarque peut également concerner les segments. On doit toutefois tenir compte du fait, admis par hypothèse, que l'évaluation de la ressemblance dépend, en partie au moins, de l'ensemble de la suite d'échantillons, et non de propriétés locales à ceux-ci. De sorte que, l'algorithme d'évaluation ne peut être réduit à une simple itération sur les échantillons, mais doit comporter un traitement global. Plus précisément, et c'est le cas de la comparaison dynamique, on peut distinguer une évaluation échantillon par échantillon puis un calcul global visant à établir le score de ressemblance indépendamment de la différence éventuelle de longueur (nombre d'échantillons) entre le segment et le modèle.

On a vu que les P.E.R. créés par l'occurrence d'un même segment peuvent être exécutés en parallèle. Le degré de **parallélisme effectif**, et donc les performances, dépendront essentiellement des ressources qui peuvent être partagées entre les P.E.R. et de la manière, pour l'implémentation matérielle, d'assurer ce partage.

V-2.5. Synchronisation

D'après les relations qui ont été décrites entre les différents processus (cf. V-2.3.), deux phases de synchronisation apparaissent :

- création par P.I. des P.E.R. et de P.A. avec transmission aux P.E.R. du segment occurrent.
- destruction des P.E.R. et transmission de leur score à P.A.

Il conviendrait d'ajouter la destruction de P.A. et la transmission à P.I. de ses résultats. Notons qu'on pourrait également considérer que P.A. et P.I. correspondent à un seul et même processus "maître" qui crée les P.E.R. et les détruit après en avoir reçu les résultats. Toutefois, la distinction qui a été introduite permet de déboucher éventuellement

sur un recouvrement des tâches par rapport aux occurrences successives de segments. Par exemple, P.A. effectuerait en fait la fonction d'arbitrage relative au segment k-1 simultanément aux activations de P.I. et des P.E.R. relatives au segment k. Naturellement, un tel mode de fonctionnement ne peut être envisagé qu'en tenant compte de l'implémentation et, en particulier des ressources en processeurs et du temps d'exécution des processus.

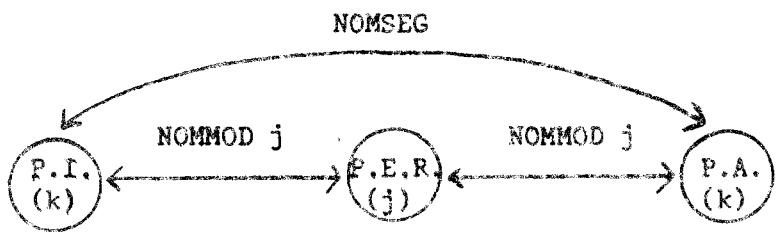
V-2.5.1. *Communications depuis le processus d'initialisation*

La création des P.E.R. est donc confiée à P.I. Cette fonction suppose :

1) L'attribution d'un nom au P.E.R. Ce nom est en particulier utilisé par P.A. Comme chaque P.E.R. correspond à un modèle et à l'occurrence d'un segment, il est assez naturel de constituer le nom du P.E.R. grâce à ces deux informations. Si on appelle NOMMOD le nom d'un modèle et NOMSEG le nom de l'occurrence du segment (en pratique, l'indice k), le nom d'un P.E.R. est de la forme :

(NOMMOD, NOMSEG)

Cependant, on peut observer le système entre la création des P.E.R. (et de P.A.) et leur destruction après fourniture du résultat à P.A.. Dans cet intervalle, qui correspond à une unité d'activité du système, on peut admettre que celui-ci ne contienne que les P.E.R. relevant d'une même occurrence k. En effet, de cette façon, on assure aisément le respect de l'ordre séquentiel des segments et on limite le nombre de P.E.R. créés, en simplifiant la synchronisation. Si cette condition est satisfaite, il n'y a aucune ambiguité à désigner simplement un P.E.R. par le nom du modèle qui lui correspond Dans ce cas, le nom du segment NOMSEG n'est connu explicitement que par P.I. et P.A. Le nom NOMMOD sert d'intermédiaire entre P.I. et les P.E.R. d'une part (création des P.E.R.), les P.E.R. et P.A. d'autre part (destruction des P.E.R.). On a donc, du point de vue des noms :



Il s'agit là uniquement d'indiquer les noms par lesquels les processus susceptibles de communiquer se connaissent mutuellement. Compte-tenu de la remarque faite plus haut sur un recouvrement éventuel des fonctions des P.E.R.

et de P.A., à un niveau d'observation donné, coexisteraient le P.A. associé au nom NOMSEG k et les P.E.R. implicitement associés au nom NOMSEG k+1.

2) La transmission au P.E.R. des informations qui lui sont nécessaires. En se référant à la description faite en V-2.4, on peut admettre que celles-ci permettent au P.E.R. d'accéder aux représentations dont il a besoin. Notons qu'on peut distinguer deux types d'informations :

- information partagée entre les P.E.R. : c'est essentiellement le cas de SEGMENT, la représentation du segment occurrent.
- information propre au P.E.R. : c'est naturellement le cas de MODELE, et des informations de contrôle éventuelles. En ce qui concerne CALCUL, qui indiquera l'algorithme d'évaluation, il semble plus général de considérer également qu'il s'agit d'une information propre à chaque P.E.R. (ce qui ouvre la possibilité d'un traitement de type MIMD).

La diffusion à tous les P.E.R. de SEGMENT, en lecture, ne pose pas de difficulté importante : il faut néanmoins assurer la disponibilité de SEGMENT aux P.E.R. tant qu'il subsiste l'un d'eux qui n'y a pas accédé. (Cela relève d'un schéma classique "producteur-consommateur"). En pratique comme la création d'un P.E.R. est conditionnée par la disponibilité d'un nouveau segment, on doit permettre cet accès aux P.E.R. en premier lieu. De la sorte, le parallélisme possible au niveau des P.E.R. dépend en partie des possibilités de partage de l'accès à SEGMENT. Les contraintes d'utilisation de cet accès semblent a priori peu pesantes et sa mise en oeuvre paraît peu problématique.

Les informations propres à chaque P.E.R. doivent, par contre, être traitées différemment. Dans le cas précédent, on avait émission par P.I. d'une information multi-destinataires. Ici les informations qui sont communiquées s'adressent à un destinataire particulier :

- MODELE dépend directement du nom du P.E.R. qui doit l'utiliser. On peut donc considérer qu'étant désigné par NOMMODj, le P.E.R.(j) sait qu'il doit accéder à MODELE j. Etant donné le caractère statique (à l'initialisation ou à l'apprentissage près) des modèles, cet accès est pratiquement indépendant de P.I., une fois le P.E.R. créé.

- CALCUL et les éventuelles informations de contrôle dépendent naturellement de l'algorithme qui supervise le travail du système et, le cas échéant, des résultats obtenus aux occurrences précédentes. L'élaboration de cette information pour chaque P.E.R. est donc à charge de P.I., ou encore, du processus "père" qui, à travers P.I., contrôle le système. De façon générale, on peut assimiler CALCUL à une "macro-commande". Elle servira de nom d'accès à la représentation codée (programme, micro-programme, ...) de l'algorithme d'évaluation (c'est-à-dire : dans l'exemple de V-2.4., à la procédure DIST). Si nécessaire, CALCUL pourra désigner aussi les informations de contrôle, et, par exemple, signaler à un P.E.R. une modification locale du traitement voire une destruction conditionnelle à vérifier (Si CONDITION alors ECHEC = VRAI fsi). De façon analogue au cas de MODELE, on doit distinguer l'acquisition de la commande CALCUL et l'accès effectif au flux d'instructions que CALCUL permet. Toutefois, le contenu de CALCUL est déterminé par P.I. (ou son "père") de façon dynamique, alors que, on l'a vu, un P.E.R. donné doit accéder implicitement à un MODELE bien défini.

En résumé, les informations communiquées par P.I. au P.E.R. de nom NOMMODj sont :

. SEGMENT (partagé entre tous les P.E.R.)

. CALCUL qui désigne la "commande" de P.E.R. (y compris des informations de contrôle)

et implicitement, MODELE j, puisque la désignation de P.E.R. lui assigne l'un des modèles.

V-2.5.2. *Communications avec le processus d'arbitrage*

Elles ont lieu à deux niveaux distincts, en provenance de deux sources différentes :

1) A la **création de P.A.**, comme il est nécessaire de disposer d'un P.A. à chaque occurrence de segment, on peut associer NOMSEG à P.A. Toutefois, comme les P.E.R., P.A. doit recevoir les directives de la part de P.I. ; P.A. n'a pas besoin d'accéder aux représentations SEGMENT et MODELE j ($0 \leq j \leq p-1$). Par contre, il est nécessaire, avant toute acceptation de résultat émis par un P.E.R., d'indiquer à P.A. la manière de mener l'opération globale de choix sur les résultats des P.E.R.. On considèrera que (parallèlement à CALCUL, pour les P.E.R.) une "commande"

OPT permet de transmettre cette information de P.I. vers P.A.(exemple : OPT peut contenir MAX, 3 (resp. MIN, 3) ce qui signifiera qu'il faut déterminer les trois plus grands (resp. plus petits) résultats fournis pour les P.E.R. (i.e. SCORE)). Le décodage d'une telle commande se traduit pratiquement par le recours à un algorithme de tri décroissant (resp. croissant). Nous appellerons CHOIX, la procédure (ou le programme, ou le microprogramme, ou l'instruction, ...) désignée par OPT (cf. DIST désigné par CALCUL).

2) Les P.E.R. doivent transmettre à P.A. leurs résultats pour qu'il mène à bien la tâche qui lui est confiée. On se trouve alors dans une situation presque symétrique au cas de la diffusion par P.I. aux P.E.R. de SEGMENT. On avait un producteur et p consommateurs ; on a ici p producteurs et un consommateur. Cependant, si dans le premier cas, il suffisait d'assurer la disponibilité de SEGMENT aux p P.E.R. indépendamment de leur identité, il faut ici que P.A. connaisse l'identité de chaque P.E.R. émetteur. En fait, l'information émise par les P.E.R. à destination de P.A. consiste simplement en un couple : (NOMMOD, SCORE). Le destinataire d'une telle information étant nécessairement P.A., on pourra utiliser, dans l'implémentation, ce fait, en privilégiant les possibilités de communication à ce niveau, puisque le destinataire est unique et connu implicitement. Notons, toutefois, comme on l'a déjà remarqué, que l'exécution complète de la fonction confiée à P.A. suppose qu'il ait reçu (éventuellement de manière asynchrone) l'ensemble des couples (NOMMOD, SCORE). Pour éviter en partie un blocage à ce niveau, il a été prévu, dans la description des P.E.R. (V-2.4.) la substitution au résultat SCORE d'un booléen ECHEC signalant à P.A. le rejet pour le segment courant du modèle correspondant. Un usage approprié de cette technique peut d'ailleurs permettre de simplifier la tâche de P.A.

A ces deux émissions d'informations vers P.A., il convient naturellement d'ajouter les résultats fournis par P.A. Sans faire d'hypothèse supplémentaire sur le destinataire de ces résultats — ce peut être P.I., ou le processus de contrôle, "père" de P.I., ou encore un processus, créé ou éveillé par P.A., chargé d'assurer la liaison avec l'extérieur du système —, il est indifférent, ici, d'admettre que ce destinataire est P.I. (il semble alors peut-être inutile de distinguer P.A. et P.I. (cf. début de V-2.5.).

La transmission de ces résultats est en fait assez simple : comme il s'agit d'un sous-ensemble des résultats fournis par les P.E.R., les éléments du "message" transmis à l'extérieur sont du même type que ces derniers. On peut symboliser cette information comme une liste, appelée CANDIDAT, qu'on associera à chaque segment. La liste CANDIDAT comprenant seulement quelques éléments (une dizaine au plus) pourra éventuellement, si l'algorithme que matérialise CHOIX le permet, être présentée sous une forme ordonnée. Les éléments de CANDIDAT seront des couples (NOMMOD, SCORE) identiques à ceux fournis par les P.E.R.

Si on impose, en outre, que toute requête de meilleure ressemblance doit fournir un résultat, on peut interdire le cas où la liste CANDIDAT serait vide. Autrement dit, l'éventualité dans laquelle le processus P.A. aboutirait à un échec (par exemple, tous les P.E.R. ont transmis un booléen ECHEC) devra être considérée comme une anomalie et traîtée comme telle.

V-2.5.3. *Récapitulation des communications*

On a vu que l'essentiel des communications inter-processus prend place dans deux phases de synchronisation, l'une effectuée par P.I. l'autre par P.A.. Dans P.I., on peut schématiser l'éveil des P.E.R. et de P.A. de la façon suivante :

```
EVEIL : Si SEGMENT (NOMSEG) disponible alors
            créer P.A. (NOMSEG) ;
            pour NOMMOD de 0 à p-1 faire
            créer P.E.R. (NOMMOD)
                                fait ;
        fsi
```

Dans P.A., la synchronisation des P.E.R. peut être décrite ainsi :

```
COLLECTE : Recevoir (NOMSEG, OPT) ;
           Pour NOMMOD de 0 à p-1 faire
                recevoir (NOMMOD, SCORE) ou (ECHEC)
                                fait ;
```

On n'a évidemment introduit aucune séquentialité dans les communications P.E.R. → P.A., celles-ci pouvant avoir lieu sous des formes très diverses En outre, P.A. grâce à l'information OPT, transmise par P.I., peut par ailleurs accéder à la "procédure" CHOIX, comme CALCUL permet d'accéder à DIST dans les P.E.R. (cf. V-2.4.).

De cette façon, on peut résumer schématiquement l'essentiel des **informations communiquées** entre processus (fig. 27). On peut leur adjoindre des informations nécessaires à chaque processus pour son exécution mais accédées plus indirectement. Cette distinction entre informations communiquées effectivement entre processus de façon dynamique et informations obtenues dans le "contexte" de la tâche (c'est-à-dire d'une façon plus statique) peut s'avérer exploitable lors de l'implémentation.

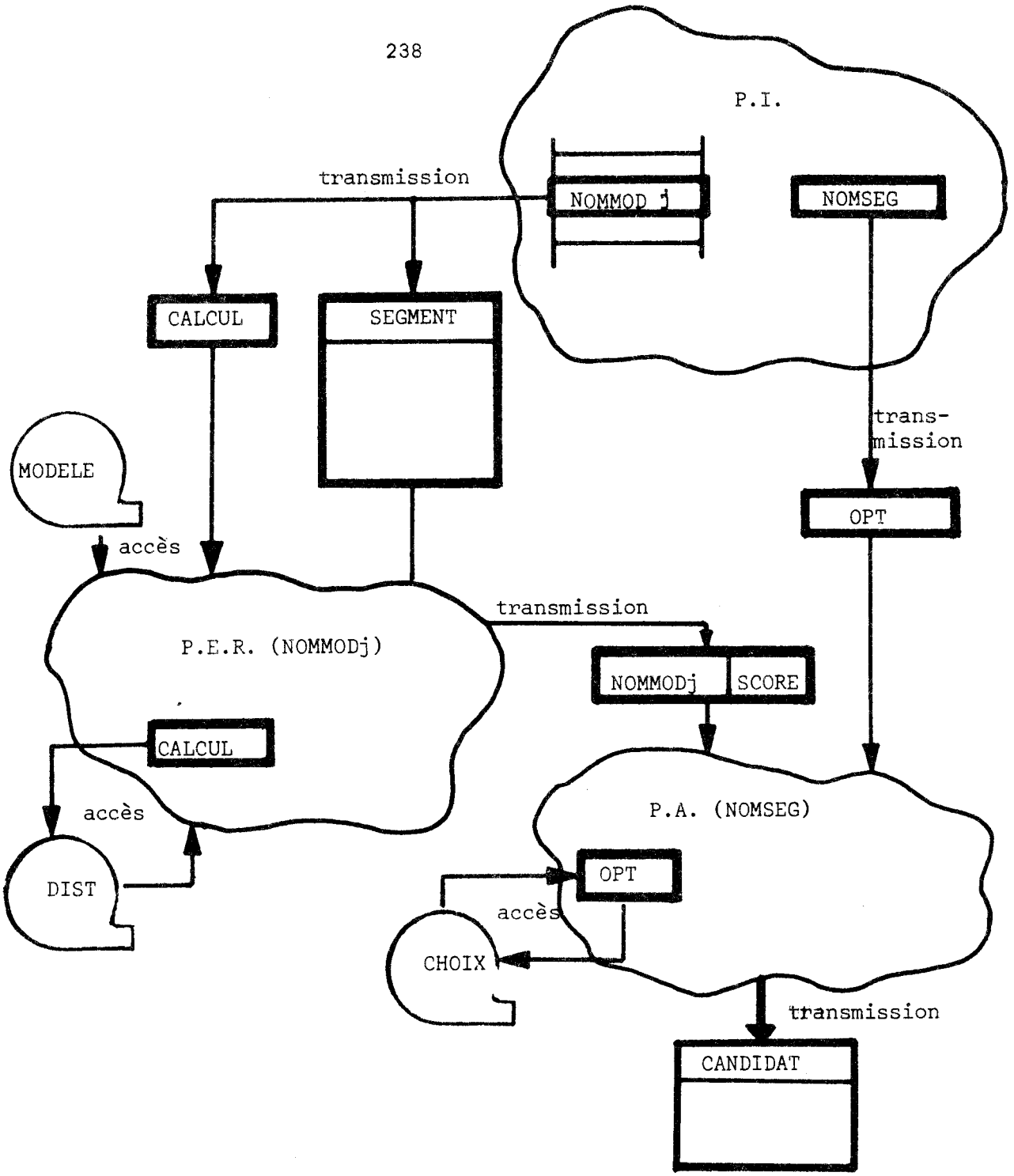


Fig. 27 : Communications entre les processus

Remarques : - Pour des raisons évidentes de simplification, on n'a figuré qu'un seul P.E.R.

- Les symboles  représentent des informations stockées séquentiellement de manière statique. Les autres informations au contraire présentent un caractère dynamique lié à chaque occurrence de segment.

- A la différence des autres informations (en général) SEGMENT est partagé entre tous les P.E.R.

V-3 - IMPLÉMENTATION PROPOSÉE

Le système, tel qu'il a été décrit, pourrait aisément être mis en oeuvre sur un processeur classique. Toutefois, un certain nombre de points évoqués dans la présentation des tâches envisagées (V-1.) semblent justifier le recours à une architecture spécifique : le plus critique semble être le **temps de réponse**, qui, à lui seul, motive l'utilisation d'un degré de **parallélisme** assez important.

V-3.1. Architecture générale

Le système proposé se compose de trois sous-ensembles essentiels (figure 28) :

1) **Le processeur d'interface et de contrôle (PIC)**
2) **Le réseau d'évaluation et de stockage (RES)**
3) **Le processeur d'arbitrage (PAR)**

Le réseau d'évaluation et de stockage, RES, est un arrangement linéaire de **cellules de mesure (CM)**. Chacune d'elle correspond à un des modèles traités. On a donc autant de CM que de modèles (i.e. un nombre relativement limité (32 à 64)).

En fait, tout se passe comme si, à chaque processus défini plus haut (voir V-2), on allouait un processeur spécifique. Cette approche peut paraître a priori excessive : cependant, elle va dans le sens des organisations distribuées évoquées au chapitre I, et un certain nombre d'avantages permettent d'étayer cette démarche :

- le temps de réponse espéré est satisfaisant : en effet, on exploite ainsi au maximum le parallélisme exprimé plus haut (V-2.2.). Le coût qui en résulte n'est pas vraiment prohibitif : en effet, la part du coût matériel dans un système baisse de plus en plus (progrès technologiques) sur le plan général. On peut alors substituer efficacement à une machine universelle "classique" un assemblage de processeurs de puissance bien moindre, mais permettant une répartition parallèle des fonctions de traitement. C'est évidemment le contexte particulier d'une application qui peut permettre cette approche.

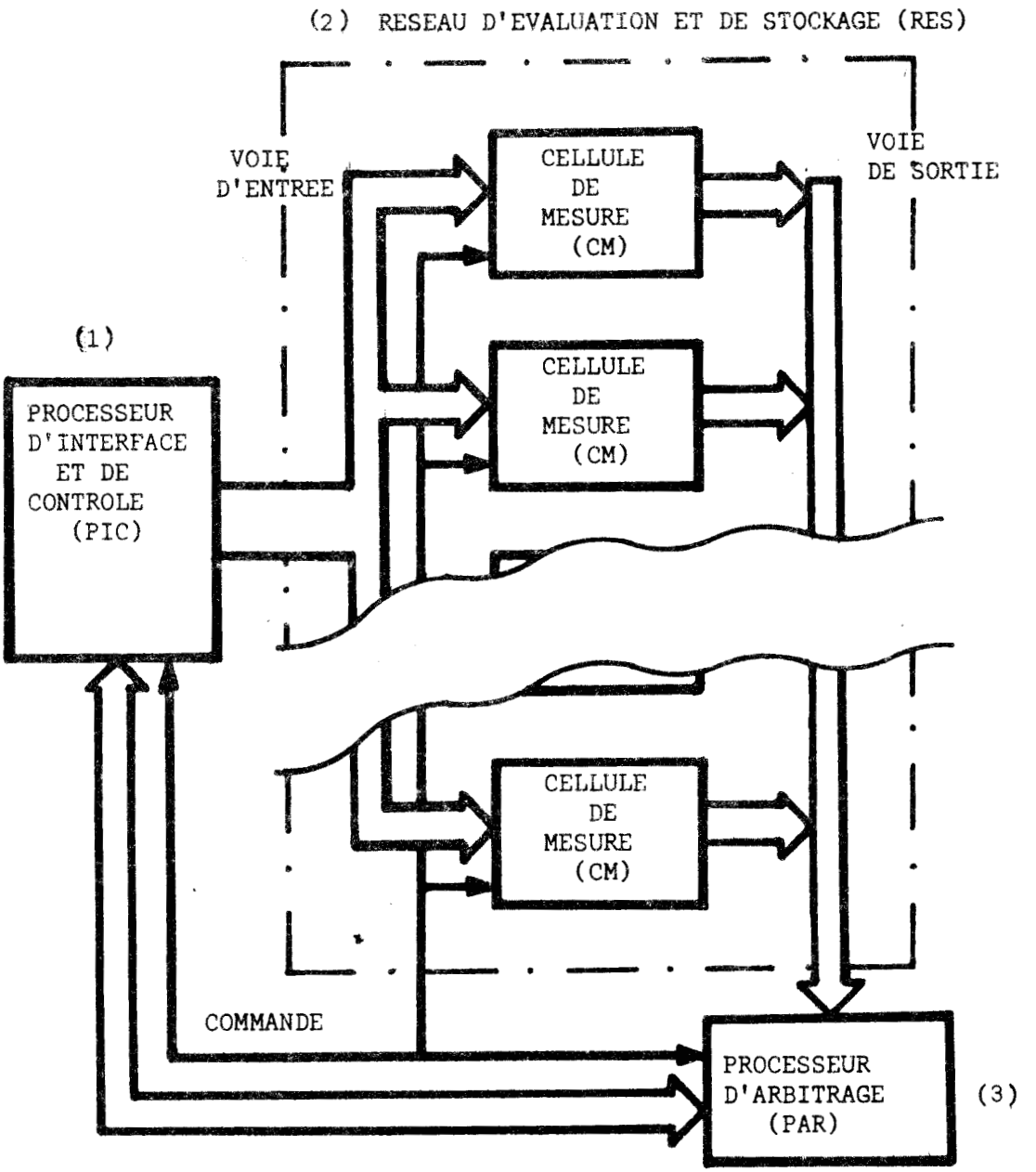


Fig. 28 : Synoptique de l'architecture proposée

- la synchronisation des processus se trouve simplifiée. En effet, si l'on avait plusieurs processus P.E.R. par processeur, le processus d'arbitrage reposerait sur deux niveaux :

. d'une part, élaboration par CM d'un résultat local,

. d'autre part, élaboration par P.A.R. du résultat final à partir des précédents.

Cela supposerait donc une tâche supplémentaire au niveau de CM. Au contraire, en adoptant le principe d'un P.E.R., c'est-à-dire un modèle, par CM, on peut obtenir un **recouvrement** effectif entre l'évaluation relative à un segment et l'arbitrage correspondant au segment précédent. De plus, la gestion des **noms de modèles** se trouve facilitée puisqu'à un modèle correspond une cellule de mesure.

- la gestion de la mémoire est également facilitée. En effet, en dotant chaque CM d'une **mémoire locale** pouvant contenir la représentation de chaque modèle, chaque P.E.R. peut y accéder de façon autonome. On évite donc des difficultés liées aux conflits d'accès.

- l'indépendance physique du traitement de chaque modèle permet aisément de pouvoir spécifier, éventuellement, un traitement adapté à chacun.

- la structure obtenue est facilement **extensible** puisqu'il suffit d'ajouter une CM pour pouvoir gérer un modèle supplémentaire et ce, sans accroissement sensible du temps de réponse (au processus d'arbitrage près). En outre, sa modularité permet une mise au point progressive et, le cas échéant, se prêtera peut-être aux techniques d'intégration - ce point sera réexaminé plus loin -.

On peut donc considérer qu'il s'agit d'une architecture multi-microprocesseurs dans laquel PIC et PAR jouent un rôle particulier et seront donc de nature différente des CM. Celles-ci, en nombre égal (ou supérieur, pour des motifs de tolérance aux pannes, le cas échéant) au nombre de modèles que comporte le problème envisagé, comporteront un espace de mémoire locale et des possibilités de traitement adaptées à la tâche d'évaluation de la ressemblance.

V-3.2. Mise en oeuvre de l'évaluation de la ressemblance

La structure qui vient d'être décrite reste trop générale si l'on ne peut préciser davantage les caractéristiques des éléments qui la composent, en particulier des cellules de mesure (CM). Toutefois, compte-tenu des diverses approches possibles (V-1), il faut préciser avant tout, sinon les algorithmes envisagés - ce qui serait trop restrictif -, du moins les incidences des techniques d'évaluation sur le matériel.

V-3.2.1. Niveau de couplage en entrée

Dans la description des communications vers les P.E.R. (V-2.5.1.) on a distingué d'une part des données statiques (modèles) et dynamiques (segment), d'autre part des "commandes".

Il faut donc tenir compte de la structure des informations, composées de suites d'échantillons.

Pour un P.E.R., c'est-à-dire pour une CM, la tâche à effectuer peut se résumer :

DIST (MODELE, SEGMENT)

DIST étant désigné par l'information CALCUL,et SEGMENT étant fourni explicitement (MODELE est connu implicitement puisqu'une CM accède localement à un modèle et un seul).

Deux niveaux de couplage entre PIC et les CM peuvent être envisagés :

1) Au niveau des échantillons : le traitement DIST est programmé sur PIC qui n'a qu'à envoyer des instructions pour chaque échantillon aux CM. Dans ce cas, on peut considérer que la charge d'un P.E.R. se trouve répartie entre PIC et une CM.

2) Au niveau des modèles : cela correspond à la description logique des P.E.R. Les informations communiquées sont plus globales en ce qui concerne le traitement. C'est à la CM de traduire localement la "commande" CALCUL.

Un certain nombre de critères permettent de préférer cette deuxième solution :

. les communications PIC→ CM sont moins volumineuses, d'autant que les instructions, au contraire des échantillons, peuvent être spécifiées différemment pour chaque CM.

. la prise en charge complète de la tâche d'un P.E.R. par la CM, au prix d'une puissance supérieure de celle-ci, correspond mieux au découpage parallèle du problème. En effet, dans le cas contraire, PIC devrait gérer les états de tous les P.E.R. simultanément.

. le modèle est l'unité "sémantique" du problème. On a donc une machine associative à logique distribuée (I-2.2.1.) analogue à ALAP. La différence réside dans le niveau d'observation, puisque le modèle joue ici le rôle du "mot-mémoire". En fait, si globalement on peut affirmer que DIST (SEGMENT, MODELE) correspond à un traitement de type SIMD, à un niveau plus fin on pourra éventuellement observer un fonctionnement MIMD.

. l'aspect hétérogène, éventuellement, des modèles peut permettre une interprétation locale spécifique de la commande émise par PIC.

. le caractère banalisé de chaque échantillon implique la répétitivité du traitement effectué sur chacun. Il serait donc inutile que PIC spécifie à chaque fois une instruction.

. le contrôle par PIC du déroulement d'un P.E.R. nécessiterait des communications bi-directionnelles entre PIC et chaque CM, ce qui compliquerait la structure matérielle. En effet, en dépit des possibilités offertes au concepteur par la baisse des coûts, les communications dans les systèmes restent un problème essentiel dans les architectures parallèles.

On suppose donc que l'on travaille au niveau "modèle". PIC transmet à chaque CM :

- une "macro-instruction" CALCUL permettant à chaque CM d'accéder à la procédure DIST.
- la suite des échantillons qui constituent SEGMENT.

V-3.2.2. *Prise en charge par la cellule de mesure*

On reprend ici l'hypothèse selon laquelle l'évaluation de la ressemblance nécessite un traitement exhaustif segment ✕ modèle (cf. V-1.3.).

La technique de comparaison dynamique constitue un exemple intéressant : on peut en fait distinguer deux phases (cf. III-2.3.5. et figure 19) :

a) calcul d'un **indice de ressemblance locale** inter-échantillons. Pour chaque échantillon du segment on évalue $d(i, j) = d(x_i, m_j)$.

Soit, pour un couple (segment, modèle), une matrice

$$D = [d(i, j)] \qquad 0 \leq j \leq N-1$$

(N = nb d'échantillons par modèle)

$0 \leq i \leq n-1$

(n = nb max d'échantillons par segment)

b) Recherche, dans le graphe dont les sommets sont pondérés par D, du **"meilleur chemin"**. Si on considère que chaque sommet a trois prédécesseurs, c'est-à-dire pour (i, j) : (i-1, j)

(i, j-1)

(i-1, j-1)

on peut calculer le résultat de proche en proche.

Plus généralement, on distinguera donc dans la tâche DIST :

- une "opération" locale à effectuer pour chaque échantillon, qu'on appellera OPL. Dans l'exemple il s'agit en fait du calcul de la ligne d(i, j) de la matrice. Le but à atteindre serait de synchroniser OPL avec le rythme de communication des échantillons par PIC.

- une "opération" globale à effectuer sur les résultats précédents qu'on appellera RESS et qui fournira le résultat SCORE.

On peut donc considérer que l'information CALCUL fournie par PIC se compose de 2 parties :

OPL	RESS

ce que la CM interprètera comme :

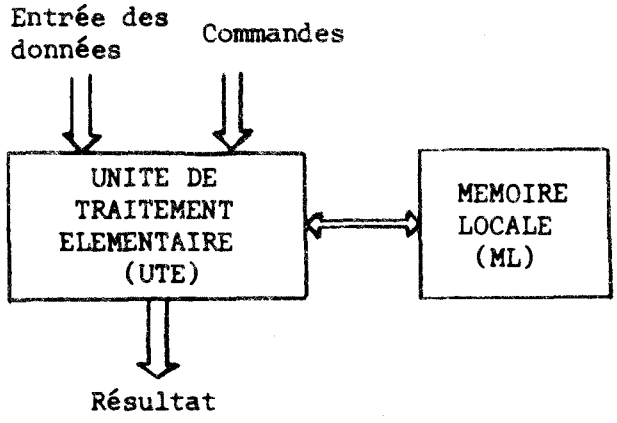
Pour i = 0 à n-1 faire OPL (éch. i) fait ;

Faire RESS ;

en supposant qu'on commande le déclenchement de OPL par l'émission par PIC de chaque échantillon.

V-3.3. Structure d'une Cellule de Mesure

D'un point de vue fonctionnel, une cellule de mesure comporte les éléments suivants :



- la voie d'entrée des données (fournies par PIC) sert, en régime dynamique, à l'acquisition des échantillons successifs d'un segment.

- la voie de commande permet au PIC d'une part d'envoyer une "macro-instruction"(OPL ou RESS), d'autre part d'émettre certains signaux de contrôle.

- la voie résultat permet à la CM d'émettre le SCORE vers PAR, en l'accompagnant, le cas échéant, d'un signal de contrôle.

- l'unité de traitement élémentaire va interprêter les commandes (par programme, micro-programme ou décodage, suivant le cas) pour effectuer des opérations portant

  . d'une part sur les couples (échantillon du segment, échantillon du modèle) -pour tout le modèle -,

  . d'autre part, sur les résultats des opérations précédentes.

- la mémoire locale répond à trois types de besoins :

  . stockage de la représentation du modèle.

  . stockage de données temporaires (espace de travail).

  . stockage d'instructions et/ou de micro-instructions.

Compte-tenu de leurs caractéristiques différentes, on aura avantage, au moins dans une phase opérationnelle de la machine, à séparer physiquement ces trois espaces-mémoire.

On peut alors détailler, en fonction des besoins et des modes de fonctionnement souhaités, la structure interne d'une cellule (figure 29). En fait, il s'agit d'une proposition, choisie comme une des possibilités de prise en charge de la description fonctionnelle.

Globalement, cette structure est à peu près celle d'un micro-processeur, à plusieurs différences près, liées au caractère spécialisé du système envisagé :

- la cellule de mesure comporte un accès à une **mémoire séquentielle électronique** permettant le stockage des modèles ou plutôt du modèle pris en charge par la cellule.

- les commandes peuvent être décodées directement, ou être interprétées par l'intermédiaire de la mémoire de programme (en fait, dans une version opérationnelle du système, celle-ci peut être une mémoire morte de micro-programmes).
- pour accélérer la première phase du traitement (c'est-à-dire : couples (échantillon du modèle, échantillon du segment)) les deux registres peuvent être chargés simultanément, l'un depuis le tampon d'entrée, l'autre depuis la mémoire séquentielle, puisque les deux flux sont physiquement distincts. En outre, il est possible, après décodage de l'instruction d'avoir préparé l'adresse destination de l'opération simultanément à ce chargement.

L'unité de traitement élémentaire* est une machine à registres qui doit offrir des possibilités **arithmétiques et logiques** indispensables au type de problème traité. En particulier, outre les opérations de transfert, des primitives telles que l'addition, la soustraction, le OU exclusif bit-à-bit permettent, par exemple, des fonctions telles que la comparaison dynamique. En fait, cette U.T.E., fonctionnellement, est peu différente d'une unité arithmétique et logique "classique". On peut cependant admettre que dans tel contexte spécifique, une opération particulière puisse être adjointe de façon câblée (par exemple, distance de HAMMING). De même, à supposer que la comparaison dynamique puisse être utilisée de façon exclusive, une primitive importante (voir V-3.2. et III-2.3.5.) consiste, pour chaque sommet, à comparer les résultats partiels, c'est-à-dire les scores affectés aux sous-chemins aboutissant à ses

*U.T.E.

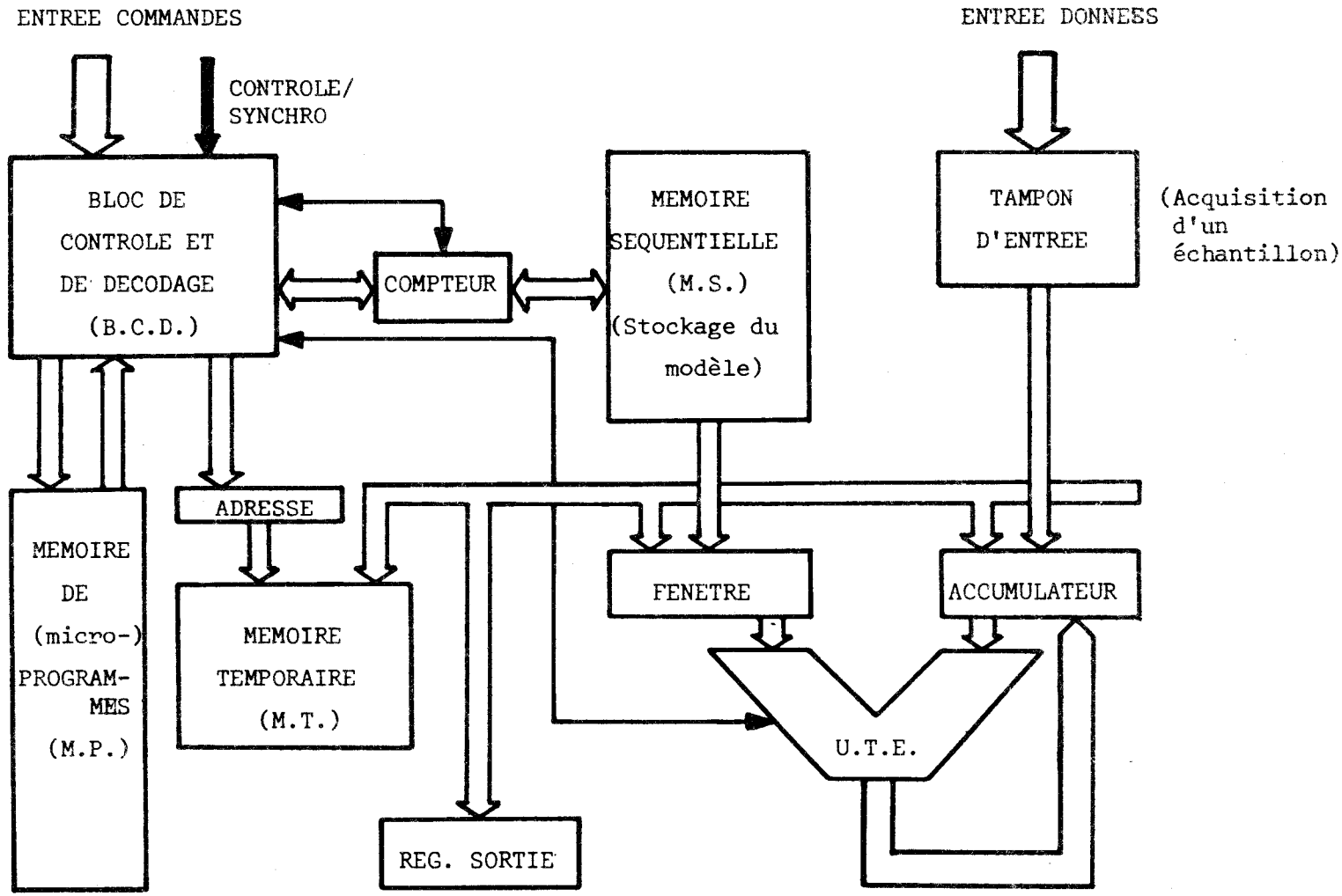


Fig. 29 : Structure interne d'une cellule de mesure

Remarque : On n'a représenté que les liaisons essentielles, en particulier en ce qui concerne les signaux de contrôle émis par le B.C.D.

3 prédécesseurs : on peut imaginer d'organiser la machine en conséquence, en prévoyant, par exemple, un comparateur à 3 comparandes.

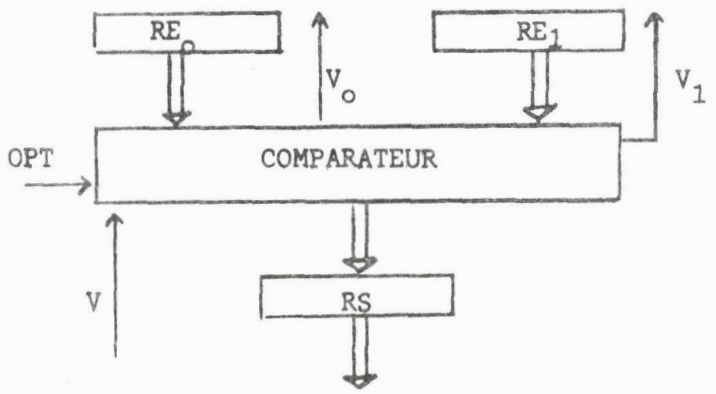
En fait, la cellule de mesure traduit, par sa capacité mémoire et ses possibilités de traitement, les caractéristiques propres à l'application.

V-3.4. Processeur d'arbitrage

On a vu (I-2.3.) l'importance, dans une machine associative, de la fonction de sélection des réponses. Comme les CM fournissent chacune un résultat SCORE ou un booléen signalant l'échec de l'évaluation, il s'agit de déterminer les noms des modèles jugés optimaux. Le critère de choix est obtenu grâce à la transmission au processeur d'arbitrage de la commande OPT (voir V-2.5.3.).

Compte-tenu des solutions envisageables, on peut distinguer deux types d'approches :
- l'une consiste à prévoir un **circuit câblé** pour prendre en charge cette fonction.
- l'autre consiste à confier le travail à un **processeur** à part entière.

Un **sélecteur arborescent** du type proposé pour ANDERSON pourrait convenir (voir I-2.3.). Les feuilles correspondraient aux registres de sortie des CMs. Un noeud d'un tel réseau pourrait alors être du genre suivant :



Cependant, un certain nombre d'inconvénients apparaissent :

1) Il s'agit d'un procédé **peu souple**, car il faut pouvoir prendre en charge de façon câblée plusieurs critères de comparaison.

2) Bien que rapide, ce procédé est adapté à la recherche du meilleur candidat. Pour fournir une réponse multiple (par ex : "les 4 meilleurs scores") il faut **réitérer** le procédé autant de fois.

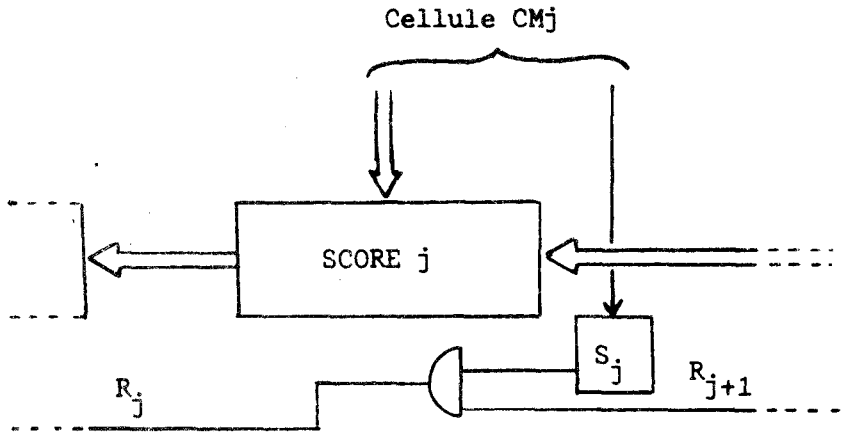
3) Du point de vue implémentation, un tel réseau est peu compatible avec les concepts de modularité et d'extensibilité mis en avant dans la structure du réseau des cellules de mesure.

Une **solution linéaire et séquentielle** paraît donc mieux adaptée :
- elle permet de respecter la **périodicité** du système, ce qui facilite son implémentation.
- le temps de réponse, meilleur dans le cas arborescent, devient une préoccupation secondaire si l'on admet le principe du recouvrement (V-3.1.) entre évaluation d'un segment et arbitrage relatif au segment précédent.

On peut ensuite observer qu'un système de sélection séquentiel câblé serait complexe à mettre en oeuvre, toujours en raison du caractère multiple des critères de sélection envisageables. Inversement, une prise en charge, par les CMs elles-mêmes, de cette fonction, grâce à leurs possibilités de traitement, alourdirait inutilement leur charge : on a vu qu'il était souhaitable du point de vue du débit du système, qu'une fois fourni le score obtenu par un segment, une CM puisse se consacrer au segment suivant.

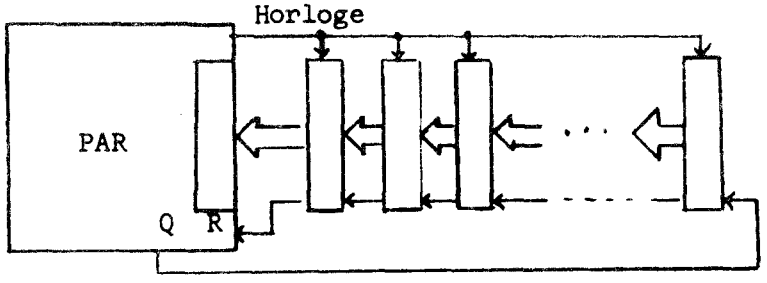
Toutes ces motivations concourent donc à un **compromis** :
- arbitrage assuré par un **processeur spécifique, PAR.**
- **communication des scores à P.A.R.** par un procédé **séquentiel**, si possible indépendant du fonctionnement interne des CMs.

L'opération se déroule donc à 2 niveaux : chaque CM dispose en sortie d'un registre SCORE et d'une bascule, S, permettant de signaler la fin de la tâche d'évaluation. Une fois le registre SCORE chargé et la bascule S positionnée, la CM n'est plus concernée.



Notons que le choix du mode de transfert, série ou parallèle, reste ouvert, en fonction des contraintes de vitesse et d'implémentation (nombre de liaisons notamment).

En bout de chaîne, se trouve le processeur d'arbitrage, PAR. Il reçoit les contenus des registres SCORE j, à condition que le signal R lui soit parvenu, auquel cas il aura émis les signaux d'horloge permettant le décalage de l'ensemble.



Le signal Q, émis par PAR permet d'initialiser la propagation du signal R_j. En fait, lorsque R parvient à PAR, toutes les CM ont terminé leur évaluation et PAR obtient leur score par décalages successifs. Un compteur, associé au décalage, permet de générer les noms relatifs à chaque score (numéro de la cellule correspondante).

La sélection revient ensuite à effectuer sur les scores un algorithme de tri, ce qui, compte du nombre relativement faible de CMs, ne pose pas de difficulté particulière.

Le cas des cellules ayant abouti à un échec (donc à éliminer de la sélection) est assez simple à traiter : on peut prévoir un bit du registre SCORE pour signaler s'il y a échec ou non. Une variante serait qu'une CM se trouvant dans ce cas émette un SCORE "asymptotique" c'est-à-dire qui l'exclue de la sélection. Par exemple, si les scores doivent être sélectionnés par ordre décroissant (recherche du MAX) on fournirait un SCORE nul.

La solution précédente présente l'inconvénient de faire attendre PAR tant que la dernière des CMs n'a pas signalé l'achèvement de sa tâche.

On pourrait alors imaginer une communication du type bus, le signal S_i fonctionnant comme une interruption pour PAR. Celui-ci pourrait donc accéder aux scores au rythme de leur émission et donc élaborer progressivement la sélection. Dans ce cas, toutefois, il faudrait une communication CM ⟷ PAR plus complexe (identification de CM, en particulier), ce qui alourdirait la tâche des CMs. En outre, le parallélisme inhérent à l'évaluation serait compromis par le traitement successif de chaque interruption. Il est donc plus simple qu'un signal unique prévienne PAR que tous les scores sont disponibles. La perte de temps subie sera d'autant plus légère que les processus d'évaluation effectués dans chaque CM seront de durée comparable.

V-3,5. Processeur d'interface et de contrôle

Dans la description précédente, on a peu parlé du processeur PIC, assurant l'interfaçage et le contrôle de la machine associative.

En fait, il peut s'agir d'une machine relativement classique, ses fonctions étant essentiellement :

- supporter le **dialogue** avec le système-hôte ou avec l'utilisateur (prise en charge d'un langage)
- **piloter la machine associative**, en lui transmettant les données (échantillons) et les commandes associées, ainsi que les différents signaux de contrôle.

Une caractéristique de ce processeur réside également dans sa fonction de tampon éventuel : en effet, c'est PIC qui va cadencer l'envoi des échantillons aux CM, en fonction du temps moyen nécessaire à l'évaluation d'un segment. Il peut donc s'avérer nécessaire de stocker temporairement un ou des segments, l'objectif étant, rappelons-le, d'aboutir approximativement au temps réel.

Le mode de fonctionnement du processeur d'arbitrage qui a été adopté (V-3.4.) peut éventuellement permettre de confondre, physiquement PIC et PAR en un seul processeur. Ce processeur gèrerait le flux incident en transmettant commandes et échantillons aux CMs. En cas d'interruption, il effectuerait la tâche d'arbitrage relative au segment précédent. Cette possibilité est évidemment conditionnée par les différents temps mis en jeu. Dans ce cas, PIC pourrait en outre prendre en compte des règles contextuelles (probabilités de succession des segments).

V-3.6. Discussion

Le modèle qui a été présenté reste relativement général. La définition d'un cahier des charges permettrait de préciser :
- les performances souhaitées,en particulier au niveau du temps de réponse
- les caractéristiques telles que taille mémoire, primitives indispensables, etc ...

Ces points influent particulièrement sur la construction des cellules de mesure.

On a surtout misé sur une représentation exhaustive, brute, des modèles et une évaluation de la ressemblance du type comparaison dynamique.

Toutefois le modèle proposé conviendrait également à une approche par extraction de paramètres, à condition que le nombre et la complexité de ceux-ci ne permettent pas la mise en oeuvre d'un algorithme sélectif, par exemple de classification ou de hash-code. Un modèle est alors considéré comme un ensemble de paramètres (nombreux). La différence essentielle par rapport au modèle décrit résiderait en un transfert de la tâche depuis les CMs vers PIC. Celui-ci serait chargé de l'extraction des paramètres sur les segments incidents, c'est-à-dire de leur codage dans une représentation homogène avec les modèles. Au niveau de ceux-ci,

en supposant que les paramètres soient pertinents, un comptage des paramètres ressemblants pourrait suffire (en particulier dans le cas où leur nombre serait assez élevé).

En ce qui concerne les étapes permettant d'envisager une réalisation, on peut faire un certain nombre d'observations :

Tout d'abord, une **simulation** pourra présenter un intérêt non négligeable. Ses hypothèses seront essentiellement le débit des informations incidentes et la longueur de la représentation des modèles, ainsi que leur nombre. Les renseignements qu'on pourra attendre concernent principalement les tailles mémoire, des CMs, d'une part, du PIC, d'autre part, les modes de communication, la puissance de calcul des CMs, etc ...

Ensuite, diverses approches peuvent être menées, selon le stade de développement souhaité. On considérera surtout le cas des cellules de mesure :

a) Niveau carte

. Une cellule de mesure peut être "simulée" par un **microprocesseur "standard"** doté de mémoire vive et de circuits d'interface. Les organes fonctionnels décrits en V-3.3., par exemple, la mémoire séquentielle et le compteur, sont alors simulés.

. Une carte prototype peut être conçue dans laquelle on réaliserait les organes décrits de façon câblée à l'aide d'**éléments discrets** existants.

Dans les deux cas, les fonctions de PIC et PAR seront confiées à une machine-hôte classique. Le but est de mettre au point la structure d'une cellule et en particulier la partie commande. Dans le second cas, toutefois, l'intérêt pourrait être d'obtenir, grâce à de la logique discrète rapide, des performances élevées.

b) Niveau intégration

Ce cas est plus hypothétique et correspondrait à une phase de développement ultérieur. Dans les possibilités actuelles on peut imaginer, par exemple :

. 3 boîtiers : 1) UTE, UCD, interface
2) mémoire vive
3) mémoire séquentielle.

1) correspondrait à peu près à une unité centrale (boitier "CPU") de microprocesseur (encore à concevoir ...)
2) supporterait l'espace de mémoire temporaire
3) comporterait les deux fonctions de mémoire de micro-programme et de stockage du modèle. Notons que, dans une phase véritablement opérationnelle, il s'agira effectivement d'une mémoire séquentielle morte (ce qui diminue le pouvoir intégrateur nécessaire).

. 2 boîtiers : cela correspond aux produits de tendance "monochip" : la mémoire morte se trouve intégrée au boîtier principal avec un espace éventuel de mémoire vive. Au besoin, un boîtier externe de mémoire vive est ajouté.

Ces hypothèses sont évidemment basées sur une certaine évaluation "moyenne" des besoins (cf. l'exemple en V-1.5). Il se peut qu'une application permette d'intégrer en un seul boîtier une cellule telle que décrite en V-3.3.. Chaque boîtier possède alors une portion de mémoire morte, comme dans les 2 cas précédents, propre à chaque modèle (représentation du modèle, micro-programmes).

On est resté, jusqu'ici,dans les limites actuelles, ou prévues, de la technologie — ou plutôt, ce sont celles-ci qui peuvent permettre d'envisager telle ou telle possibilité—. En allant plus loin (ou, ce qui est équivalent, en limitant sensiblement les ambitions de la tâche traitée) le modèle décrit pourrait se prêter à l'intégration de plusieurs cellules par boîtier.

Cependant, en ce qui concerne l'évolution de la technologie, toute prévision au-delà du court terme reste hasardeuse. Il reste néanmoins que la répétitivité de la structure proposée et son extensibilité peuvent être des atouts déterminants.

V-4 - LA RÉALISATION EN COURS

En marge de cette étude, un processeur associatif adapté au contexte du traitement de la parole a été défini. Il est en cours de réalisation.

Sa conception se rattache à la présente étude par un certain nombre d'aspects mais en diffère quelque peu, essentiellement au niveau des objectifs.

Ses principales caractéristiques sont les suivantes :

- format large de 128 bits correspondant à un échantillon de parole de 16 canaux codés sur un octet (cf. chapitre IV et § V-1.5.) : ce format induit un degré de parallélisme important et justifie de confier le traitement à 16 Processeurs microprogrammés (Signetics 8 X 300).
- Capacité de 16 k-mots ce qui peut englober 4 k-mots d'échantillons de référence, des informations de structuration et éventuellement des informations utiles aux autres niveaux de l'analyse de la parole.

Son fonctionnement repose sur 3 principes :

- traitement parallèle/mot mais de type MIMD dans la mesure où chaque processeur peut disposer d'un contrôle local (traitement différent pour chaque canal).
- cadencement par les données incidentes (fréquence d'acquisition des échantillons)
- communications inter-processeurs et avec le reste du système après synchronisation préalable par un réseau de communication spécifique.

La synchronisation et le contrôle du parallélisme suivent un modèle de diffusion et regroupement (comme en V-2) ce qui présente deux avantages :

. correspondance entre architecture et parallélisation des tâches
. absence de hiérarchie entre tâches similaires.

Les **aspects communs** avec la proposition qu'on vient de décrire sont, entre autres :

- le **type de tâches** envisagées
- le fonctionnement en **temps réel** au rythme de l'acquisition des données
- l'**architecture modulaire et parallèle**
- le **faible couplage** entre processus (faible volume de communications)
- l'organisation globale du **contrôle** et de la synchronisation
- l'aspect simple, répétitif et le nombre limité de **tâches élémentaires** (ce qui permet d'envisager un processeur relativement "figé").
- etc ...

Les différences essentielles découlent des objectifs fixés, principalement au plan des contraintes matérielles et des **échéances**. La réalisation matérielle en cours se caractérise donc par :

- un **contexte plus restrictif**, celui du traitement de la parole, ce qui permet de définir, en particulier, format, capacité, et cadence de traitement.

l'utilisation d'**outils matériels disponibles**, ce qui réduit les possibilités envisagées dans notre étude. Des choix, restés ouverts dans notre proposition ont dû ainsi être définis.

On peut noter quelques différences dans les solutions adoptées, par exemple, pour l'**outil de synchronisation** (bien que dans les deux cas, on ait tenté de parvenir à une certaine simplicité) ou la répartition des unités de traitement par rapport au format de stockage : ainsi chaque processeur "voit" un canal (pour tous les échantillons de chacun des segments phonétiques de référence). De l'autre, en supposant une adaptation exclusive de l'architecture proposée au traitement de la parole, chaque processeur "voit" un phonème.

Enfin, rappelons que l'aspect fonctionnel et logique de l'étude (chapitre II et III), bien que plus général, et l'examen du traitement de la parole (chapitre IV) sont des **bases communes** aux deux propositions. En particulier, en s'appuyant sur les points qui ont été présentés, on peut aborder la conception d'un **logiciel** et l'**intégration au système-hôte** pour la réalisation matérielle en cours.

CONCLUSION

A la fin de cette étude, on a pu mesurer ce en quoi une démarche uniquement fondée sur l'**outil associatif** visant à la résolution d'un problème, peut s'avérer délicate.

On peut alors résumer les principaux résultats de cette étude :

. Le **bilan** des travaux effectués dans le domaine a permis de mettre en évidence l'aspect spécialisé et partiel d'une solution associative, d'où la nécessité d'une **démarche de conception particulière**.

. Du point de vue de l'utilisateur (machine, processus, opérateur humain), ce sont les **manipulations** possibles de l'**ensemble des objets** stockés qui sont intéressantes. Dès lors, on a pu construire, avec un certain recul vis-à-vis de notions classiques, l'**aspect fonctionnel** d'une mémoire associative.

. Sur cette base, grâce à certaines hypothèses portant sur la représentation des objets et la caractérisation des requêtes associatives, une **organisation logique** a pu être proposée.

. Il est alors apparu que seule la double nécessité d'une **consultation exhaustive** des objets et d'un **accroissement des performances** par rapport à une solution séquentielle programmée sur une machine classique, pouvaient justifier une architecture spécifique. On a donc discuté des différents modes d'implémentation.

. Le contexte de la **reconnaissance automatique de la parole** compte-tenu de sa complexité peut profiter à plusieurs égards des apports des mécanismes associatifs. Cependant, on a pu constater, d'une part, l'**influence du système de reconnaissance choisi**, d'autre part, la **diversité des niveaux d'intervention** du traitement associatif, tant au plan des algorithmes qu'à celui de l'implémentation.

. C'est pourquoi, l'**architecture proposée** procède d'**hypothèses** à la fois **plus restrictives** — car, dans le contexte de la reconnaissance de la parole, d'autres approches seraient possibles —, et plus **générales** — dans la mesure où il semble que des domaines voisins d'applications puissent être envisagés—.

Ces différents points permettent de dégager quelques réflexions :

La description fonctionnelle, semble-t-il, pourrait servir de base à une formalisation rigoureuse (sur le plan mathématique) de la mémoire associative en tant que **système modélisable**.

D'autre part, l'étude, sous l'angle de l'organisation logique, des différentes **caractéristiques des architectures associatives** peut concourir à une meilleure appréhension du taux performances/coût ce qui améliorerait la souplesse d'utilisation de tels dispositifs.

Ces deux pôles, associés à une démarche d'utilisation adaptée à l'"outil" associatif doivent tendre à une **meilleure définition** de celui-ci et à une **disponibilité accrue** pour le concepteur d'un système. A la limite, l'objectif serait d'aboutir à la notion de **"composant associatif"** (éventuellement au pluriel), connu de façon transparente par son aspect fonctionnel, en tant que "boîte noire".

A partir des aspects matériels, et utilisant à la fois la caractérisation des requêtes associatives et le mode de représentation des objets, il devrait être possible de définir un langage adapté, voire un logiciel de base.

Sur le plan de la technologie, on peut également imaginer que certains procédés physiques soient examinés sous l'angle particulier de leur possible adéquation aux mécanismes associatifs (ex : holographie, mémoires à bulles, voire stockage analogique ...)

En ce qui concerne plus précisément la proposition d'architecture associative, plusieurs points seraient à détailler :

- meilleure définition, d'après les hypothèses adoptées, du **type de tâches** relevant de cette architecture.
- **vérification** des caractéristiques de fonctionnement, par simulation ou **réalisation** d'une maquette.
- dans ce dernier cas, on pourrait étudier l'adéquation de ce type d'architecture à un **problème réel** ce qui suppose un problème clairement identifié (cahier des charges).

Enfin, pour ce qui est de la **reconnaissance automatique de la parole**, on peut insister sur la nécessité d'une référence à une démarche complète, englobant l'**ensemble du système conçu**, et faisant appel, le cas échéant, à telle ou telle fonction associative. Etant donné la complexité d'un système de reconnaissance automatique de la parole continue, cela suppose une tout autre approche.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[1] J. MINKER : *"An overview of associative or content-addressable memory systems"*
Computing Reviews, 12, 10. oct. 71, pp 453-456.

[2] B. PARHAMI : *"Associative memories and processors : an overview and selected bibliography"*
Proc. IEEE Vol 61, 6, June 73, pp 722-730.

[3] S.S. YAU, H.S. FUNG : *"Associative processor architecture : a survey"*
Computing Surveys, Vol 9, n° 1, March 77, pp 3-27.

[4] V. CORDONNIER : *"Architecture des ordinateurs"*
Ecole d'été AFCET, Lannion, Juillet 76.

[5] M.J. FLYNN : *"Some computer organizations and their effectiveness"*
IEEE Transactions on Computers, 21, 9, Sept. 72, pp 948-960.

[6] G. SAYRE : *"STARAN : an associative approach to multiprocessor architecture"*
RAIRO Informatique, Vol 10, n° 5, mai 76, pp 59-76.

[7] G.H. BARNES, R.M. BROWN, D.L. SLOTNICK, M. KATO, D.J. KUCK, R.A. STOKES *"The ILLIAC IV Computer"*
IEEE Transactions on Computers, 17, 8, Aug. 68, pp 746-757.

[8] G. MICHEL : *"Conception et réalisation d'un processeur associatif"*,
Thèse de Docteur-Ingénieur (Université Rennes Déc. 74).

[9] C.C. YANG, S.S. YAU : *"A cut-point cellular associative memory"*
IEEE Transactions on Computers, 15, 4, Aug. 66, pp 522-528.

[10] C.Y. LEE, M.C. PAULL : *"A content-addressable distributed logic memory with applications to information retrieval"*
Proc. IEEE, Vol 51, June 63, pp 924-932.

[11] M. EDELBERG, R.L. SCHISSLER : *"Intelligent memory"*
Proceedings of the 1976 AFIPS NCC, pp 393-400.

[12] H.G. MARTIN : *"A discourse on a new super computer : PEPE"*
in "High Speed Computer and Algorithm organizations" pp 101-112
(Acad. Press, N.Y., 1977)

[13] K.J. THURBER : *"PEPE"*
in "Large scale computer architectures : Parallel and Associative
Processors" (Hayden Book, Rochelle Park, N.J., 1976), p 231.

[14] C.A. FINNILA, H.H. LOVE : *"The Associative Linear Array Processor"*
IEEE Transactions on Computers, Vol 26, 2, Feb 77, pp 112-125.

[15] G. ANDERSON, R.Y. Kain : *"A content-addressed memory designed for data base applications"*
Proceedings of the 1976 I.C.P.P.*, pp 191-195.

[16] K.J. THURBER *"STARAN"*,
in "Large scale computer architectures : Parallel and Associative
Processors" (Hayden Book, Rochelle Park, N.J., 1976), p 210.

[17] K.E. BATCHER *"The multidimensional access memory in STARAN"*
Special correspondance IEEE Transactions on Computers, 26, 2
Feb. 77, pp 174-177.

[18] T. FENG : *"Data manipulating functions in parallel processors and their implementation"*
IEEE Transactions on Computers, 23, 3, March 74, pp 309-318.

[19] H.S. STONE *"Parallel processing with the perfect shuffle"*
IEEE Transactions on Computers, 20, 2, Feb. 71, pp 153-161

[20] D.H. LAWRIE : *"Access and alignment of data in an array processor"*
IEEE Transactions on Computers, 24, 12, Dec 75, pp 1145-1155.

* *International Conference on Parallel Processing*

[21] T. LANG : *"Interconnections between processors and memory modules using the shuffle-exchange network"*
IEEE Transactions on Computers, 25, 5, May 76, pp 496-503.

[22] W.A. CROFUT, M.R. SOTTILE : *"Design techniques of a delay-line content-addressed memory"*
IEEE Transactions on Computers, 15, 8, Aug. 66, pp 529-534.

[23] R.T. RUX : *"A glass delay line content-addressed memory"*
IEEE Transactions on Computers, 18, 6, June 69, pp 512-520.

[24] D.L. SLOTNICK : *"Logic-per-track devices"*
in "Advances in Computers", Vol. 10, pp 291-296.(Acad.Press,N.Y.,1970)

[25] S.Y.W. SU : *"Cellular logic devices : concepts and applications"*
Computer,Vol 12, n°3, March 79, pp 11-25.

[26] E.F. CODD : *"A relational model of data for large shared data banks"*
CACM, 13, 6, June 70, pp 377-387.

[27] A.E. BANDURSKI *"Des mémoires associatives pour les bases de données"*
01 Informatique, n° 133, Sep 79, pp 95-100.

[28] D.C.P. SMITH, J.M. SMITH : *"Relational data base machines"*
Computer, Vol 12, n° 3, March 79, pp 28-38.

[29] P.B. BERRA, E. OLIVER : *"The role of associative array processors in data base machine architectures"*
Computer, Vol 12, n° 3, March 79, pp 53-61.

[30] G.J. LIPOVSKI : *"Semantic paging on intelligent disks"*
4th Workshop on Computer Architecture for Non-Numeric Processing, (ACM, Syracuse Univ., N.Y.) Aug. 78, p 30-34.

[31] G.J. LIPOVSKI : *"On imaginary fields, token transfers and floating codes in intelligent secondary memories"*
3rd Workshop on Computer Architecture for Non-Numeric Processing, (ACM, Syracuse Univ., N.Y.) May 77, pp 17-22.

[32] P.J. SADOWSKI, S.A. SCHUSTER : "*Exploiting parallelism in a relational associative processor*"
4th Workshop on Computer Architecture for Non-Numeric Processing (ACM, Syracuse Univ., N.Y.) Aug 78, p 99-109.

[33] C.S. LIN : "*Sorting with associative secondary devices*"
Proceedings of the 1977 AFIPS National Comp. Conference, pp 691-695.

[34] D.K. HSIAO, J. BANERJEE : "*DBC - A Data Base Computer for very large data bases*"
IEEE Transactions on Computers, 28, 6, June 79, pp 414-429.

[35] R.M. BIRD, J.C. TOU, R.M. WORTHY : "*Associative parallel processors for searching very large data bases*"
3rd Workshop on Computer Architecture for Non-Numeric Processing (ACM, Syracuse Univ., N.Y.) May 77, pp 8-16.

[36] Electronics, 1979, June 7, p 73 : "*Data-access system searches by content, slashing response time*"

[37] Informatique et Gestion, 1979, n° 111, Décembre, p 19.

[38] H. CHANG : "*Bubbles for relational data bases*"
4th Workshop on Computer Architecture for Non-Numeric Processing (ACM, Syracuse Univ., N.Y.) Aug. 78, pp 110-115.

[39] R.R. SEEBER, A.B. LINDQUIST "*Associative memory with ordered retrieval*"
IBM Journal on Research and Development, Vol 6, Jan. 62, pp 126-136.

[40] C.C. FOSTER, F.D. STOCKTON : "*Counting responders in an associative memory*"
IEEE Transactions on Computers, 20, 12, Dec 71, pp 1580-1583.

[41] G.A. ANDERSON : "*Multiple-match resolvers : a new design method*"
IEEE Trans. Comp. 23, 12, Dec 74, pp 1317-1320.

[42] Y. CHU : "A *destructive-readout associative memory*"
IEEE Transactions on Computers, 14, Aug 65, pp 600-605.

[43] C.C. FOSTER : "*Determination of priority in associative memories*"
IEEE Transactions on Computers, 17, Aug. 68, pp 788-789.

[44] H. WEINSTEIN : "*Proposals for ordered sequentiel detection of simultaneous multiple responses*"
IEEE Transactions on Computers, 12, Oct 63, pp 564-567.

[45] J.R. CARLBERG : "A *paged hardware associative memory*"
Aug. 77, (Rapport du) David W. Taylor Naval Ship Research and Development Center, Bethesda, Maryland.

[46] S.N. PORTER : "*Use of multiwrite for programmability of search memories*"
Journal of the ACM, 13, Jul. 66, pp 369-373.

[47] D. RANDET : " ... - *Nouveaux types de mémoires : études et recherches*"
Techniques de l'Ingénieur, série H1, Sept. 77.

[48] J.D. BARNARD, F.A. BEHNKE, A.B. LINDQUIST, R.R. SEEBER
"*Structure of a cryogenic associative processor*"
Proc. IEEE, Vol 52, oct 64, pp 1182-1190.

[49] J.P. PRITCHARD, L.D. WALD : "*Design of a fully associative cryogenic data processor*"
IEEE Transactions on Magnetics, 1, March 65, pp 68-71.

[50] IBM Journal on Research and Development, Vol 24, 2, March 80, "Josephson Computer Technology" (Special Issue)

[51] W.L. Mc DERMID, H.E. PETERSEN : "A *magnetic associative memory*"
IBM Journal on Research and Development, 5, Jan 61, pp 59-62.

[52] J.R. KISEDA, H.E. PETERSEN, W.C. SEELBACH, M. TEIG :
"A *magnetic associative memory*"
IBM Journal on Research and Development, 5, Ap 61, pp 106-121.

[53] W.F. CHOW : *"Plated-wire content-addressable memories with bit-steering technique"*
IEEE Transactions on Computers, 16, Oct 67, pp 642-652.

[54] J.I. RAFFEL, T.S. CROWTHER : *"A proposal for an associative memory using magnetic films"*
IEEE Transactions on Computers, 13, Oct 64, p 611.

[55] C. BATTAREL : *"Mémoires à propagation de domaines plans"*
Techniques de l'Ingénieur, série H1, Mars 77

[56] J.P. HUIGNARD, E. SPITZ : *"Mémoires optiques"*
Techniques de l'Ingénieur, Série H1, Déc. 76

[57] D. GABOR : *"Associative holographic memories"*
IBM Journal of Research and Development, 13, March 69, pp 156-159.

[58] M. SAKAGUCHI, N. NISHIDA, T. NEMOTO : *"A new associative memory system using holography"*
IEEE Transactions on Computers, 19, 12, Dec 70, pp 1174-1181.

[59] J. BOREL : *"Technologie des mémoires à semi-conducteurs"*
Techniques de l'Ingénieur, série H1, Déc 77.

[60] Electronics 1978, Janv.19, pp 7E-8E "Associative processor on a chip due from British laboratory"

[61] P.VANLAER : *"Etude d'une architecture à flux simultanés"*
Thèse de Docteur Ingénieur, Université de Lille I, Avril 1979

[62] W.M. GOSNEY *"Reappraising CCD : can they stand up to RAMs ?"*
Electronics, 1979, June 7, p 122.

[63] Minis et Micros, 1980, nº 117, 28 avril p 29.

[64] G.C. BONGIOVANNI, F. LUCCIO : *"Permutation of data blocs in a bubble memory"*
Communications of the ACM, 22, Jan 79, p 21.

[65] T.C. CHEN, C. TUNG : *"Storage management operations in linked shift register loops"*
1975 Fall IEEE Computer Conference, Sep 9-11, pp 16-18.

[66] K. TAKAHASHI : *"Symbol string pattern recognition by magnetic bubble devices"*
1975 Fall IEEE Computer Conference Sep 9-11, pp 93-96.

[67] S.Y. LEE, H. CHANG : *"Associative-search bubble devices for content-addressable memory and array logic"*
IEEE Trans. Comp. 28, 9, Sep 79, pp 627-636.

[68] S.Y. LEE, H. CHANG *"Associative-search bubble devices for content-addressable memories"*
1975 Fall IEEE Computer Conference Sep 9-11, pp 91-92.

[69] M. BACONNIER : *"Mémoires à bulles magnétiques : où en est-on aujourd'hui"*
Minis et micros, 1979 n° 108 Déc 79, p 17.

[70] Electronics : 1980, Jan 17, p 164 "FIFO's get denser, faster"

[71] A.D. FALKOFF : *"Algorithms for parallel search memories"*
Journal of the ACM, 9, 4, Oct. 62, pp 488-511.

[72] L. UNGARO : *"Techniques associatives de traitement"*,
Thèse de Docteur-Ingénieur, Université de Rennes, Déc 74.

[73] D.E. KNUTH : *"The art of computer programming"*
Vol 3 "Searching and sorting" (Addison-Wesley, 1972)

[74] J.L. BENTLEY : *"Multidimensional binary search trees used for associative searching"*
Communications of the ACM, 18, 9, Sep 75, pp 509-517.

[75] C.E. GOBLE : *"A free-text retrieval system using hash codes"*
Computer Journal 18, 1, Feb 75, p 18.

[76] W.A. BURKHARD : *"Associative retrieval trie hash-coding"*
Journal of Computer and System Sciences, 15, 1977, pp 280-299.

[77] R.L. RIVEST : *"Analysis of associative retrieval algorithms"*
Rapport du LABORIA n° 54, IRIA, Fév. 74.

[78] R.G. LANGE : *"High level language for associative and parallel computations in STARAN"*
Proc. of the 1976 ICPP (voir réf. [15]), p 170.

[79] J.A. FELDMAN, P.D. ROVNER : *"An Algol-based associative language"*
Communications of the ACM, 12, Aug 69, p 439

[80] P. BOULLIER, P. JANCENE : *"LAMBDA : Un langage pour la manipulation d'une base de données associative"*
Rapport du LABORIA N° 25, IRIA Août 73.

[81] N.V. FINDLER : *"On the role of exact and non-exact associative memories in human and machine information processing"*
Séminaire IRIA, Classification automatique et perception par ordinateur, 1973, p 215.

[82] J.W. WIRTH : *"ASTROL - An associative structures oriented language"*
Proc. of the 1979 AFIPS National Comp. Conference pp 727-732.

[83] N.J. STILLMAN, P.B. BERRA : *"A comparison of hardware and software associative memories in the context of computer graphics"*
Communications of the ACM, 20, 5, May 77, pp 331-338.

[84] A.G. HANLON : *"Content-addressable and associative memory systems : a survey"*
IEEE Transactions on Computers, 15, 4, Aug 66, pp 509-521.

[85] L.C. HIGBIE : *"Associative processors : a panacea or a specific ?"*
Computer Design, Jul 76, pp 75-82.

[86] M.W. SUMMERS : *"An associative processor application study"*
RADCAP/AWACS Technical Report, Rome Air Development Center, N.Y. Jan 76.

[87] J.H. KATZ : *"Analysis of the AWACS passive tracking algorithms on the RADCAP Staran"*
Proc. of the 1976 ICPP (voir réf. [89]), p 177.

[88] H.H. LOVE : *"Radar processing in the ALAP"*
Proc of the 1976 ICPP (voir réf. [89]), p 161.

[89] Proceedings of the 1976 International Conference on Parallel Processing Aug 24-27 1976 (IEEE, New York)

[90] Actes du 7ème Colloque International sur l'Architecture des Ordinateurs Mai 1980, La Baule.

[91] S.S. YAU, C.C. YANG :*"Pattern recognition by using an associative memory"*
IEEE Transactions on Computers, 15, n° 6, Dec 66, pp 944-947.

[92] C.H. SMITH, L.D. WITTIE : *"Memory hardware for high speed job selection"*
IEEE Transactions on Computers, 25, Feb 76 N° 2, p 148.

[93] IEEE Transactions on Computers 27, june 78, n° 6, "Fault-tolerant computing" (special issue)

[94] COMPUTER, Vol 13, n° 3, March 80, "Fault-tolerant computing" (special issue)

[95] M.M. ZLOOF : *"Query-by-example ; a data base language"*
IBM Systems Journal, 16, 4, 1977, p 324.

[96] A. MESGUICH, B. NORMIER : *"Interrogation de données en langage quasi-naturel"*
Actes du Congrès AFCET, 1976, Tome 2, p 583.

[97] D. COULON, D. KAYSER ; *"Identification des questions en langage naturel par apprentissage"*
RAIRO Informatique 11, 1, 1977, pp 75-97.

[98] R.G. CASEY : *"Design of tree structures for efficient querying"*
Communications of the ACM, 16, 9, Sep 73, p 549.

[99] C. PICARD : *"Théorie des questionnaires"*
(Gauthier-Villars, Paris, 1965).

[100] S. SPACCAPIETRA : *"Description fonctionnelle des données"*
RAIRO informatique, 77, vol 11, n° 4, p 351.

[101] J.E. EARLEY : *"Towards an understanding of data structures"*
Communications of the ACM, vol 14, n° 10, pp 617-618 oct. 71.

[102] D. BOULANGER et al. : *"Sémantique et structures de données"*
Actes du Congrès AFCET 1976, Tome 1, p 315.

[103] C. PAIR, M.C. GAUDEL : *"Les structures de données et leur représentation en mémoire"*
IRIA (juin 1977, Paris)

[104] P.PIN, S. CHEN : *"The entity-relationship model - A basis for the enterprise view of data"*
Proceedings of the 1977 AFIPS National Computer Conference, p 77.

[105] B. MEYER : *"Description des structures de données"*
Bulletin de la D.E.R., EDF, série C, n° 2, 1976, pp 81-90.

[106] CROCUS : *"Systèmes d'exploitation des ordinateurs"*
(DUNOD, Paris, 1975).

[107] L.C.W. POLS : *"Realtime recognition of spoken words"*
IEEE Transactions on Computers, 20, n° 9, sep 71, pp 972-978.

[108] J. MAKHOUL : *"Speaker adaptation in a limited speed recognition system"*
IEEE Transactions on Computers, 20, n° 9, sep 71, p 1057.

[109] H.F. SILVERMAN, N.R. DIXON : *"The 1976 Modular Acoustic Processor"*
IEEE Transactions on Acoustics Speech and Signal Processing, 25, n° 5, oct. 77, p 367.

[110] T. ICHIKAWA, K. SAKAMURA, H. AISO : "ARES - A *memory capable of associating stored information through relevancy estimation*"
Proc. of the 1977 AFIPS N.C.C., vol 46, pp 947-954.

[111] M.P. LECOUFFE : "*Etude et définition d'un modèle de machine parallèle dynamique dirigé par les données*"
Thèse 3e cycle, juillet 1979, Université de Lille I.

[112] B. PETITPREZ : "*A flexible circulating memory for communication a multiprocessor*"
Proc. Euromicro Symposium, Sep 80, London.

[113] L. MOUSSU : "*Méthodes d'analyse et de reconnaissance automatiques de la parole*",
Mémoire DEA, Université de Lille I, juin 1978.

[114] J.L. FLANAGAN : "*Speech analysis, synthesis, and perception*"
(Springer Verlag, New-York, 1972)

[115] D.R. REDDY : "*Speech recognition*"
(Academic Press, N.Y. 1975)

[116] R. DE MORI : "*Recent advances in automatic speech recognition*"
Int. Joint Conf. on Pattern Recognition, Kyoto, Avril 1978 pp 106-124.

[117] R. CARRE, J.P. HATON, J.S. LIENARD : "*Reconnaissance et synthèse de la parole - Etat de la recherche et du développement*"
Les Synthèses du SESORI, Sept. 1979 IRIA.

[118] P. QUINTON : "*Contribution à la reconnaissance automatique de la parole. Utilisation de méthodes heuristiques pour la reconnaissance de phrases*"
Thèse Etat, Université Rennes, Fév. 1980.

[119] G. MERCIER, P. QUINTON, R. VIVES : "*KEAL - Un système pour un dialogue avec une machine*"
Congrès AFCET, Nov. 1978, pp 304-313.

[120] P. QUINTON : *"Utilisation d'un analyseur syntaxique pour la reconnaissance de la parole continue"*
Annales des Télécommunications, 32, 1977, n° 9-10, pp 323-336.

[121] W.A. LEA : *"A prosodically guided speech understanding strategy"*
IEEE Trans. ASSP (voir réf [109] 23, n° 1, feb. 75, p 30.

[122] R.D. FENNELL, V.R. LESSER : *"Parallelism in artificial intelligence solving : a case study of Hearsay II"*
IEEE Transactions on Computers, 26, n° 2, Feb. 77, pp 98-111.

[123] R.J. SWAN et al : *"CM* : A multi-microprocessor computer system"*
1977 AFIPS NCC, Vol 46, pp 637-663.

[124] J. CAELEN : *"Un modèle d'oreille - Analyse de la parole continue - Reconnaissance phonémique"*
Thèse Etat, Univ. Paul Sabatier, Toulouse Juin 1979.

[125] V. CORDONNIER, B. TOURSEL : *"Architecture d'une mémoire intelligente"*
Journées "Machines Bases de Données", Sophia-Antipolis, Sept 80.

[126] G. MERCIER : *"Evaluation des indices acoustiques utilisés dans l'analyseur phonétique du système KEAL"*
9e Journées d'Etudes sur la Parole, AFCET-GALF, juin 78.