LES STRUCTURES DE L'INFORMATION

- o -

Cours de M. PAIR

- o -

ALES Juillet 71

Corrections : J.C. BOUSSARD
M. CHABRE

S O M M A I R E

CHAPITRE 1 - GENERALITES

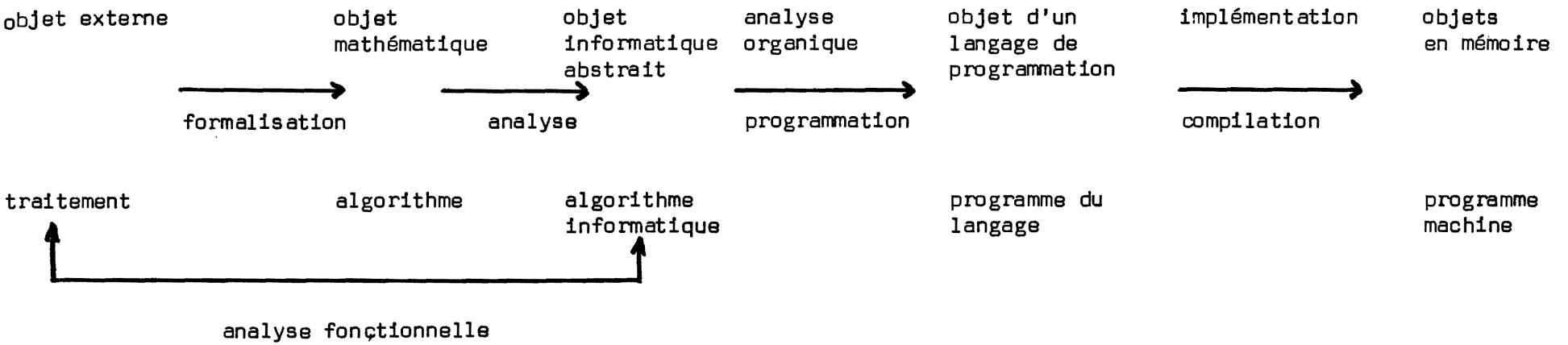
1.1.Introduction

L'informaticien a à représenter dans la mémoire d'un ordinateur des objets externes sur lesquels il veut effectuer des traitements. Il existe d'ailleurs en général des intermédiaires entre les objets du monde concret et leurs représentations en mémoire : des objets mathématiques formalisant les objets externes (par exemple des graphes), des objets manipulés par les langages de programmation (par exemple des tableaux, des structures).

objets externes → objets mathématiques → objets d'un langage de programmation → objets en mémoire

traitements → algorithmes → programmes du langage → programmes machine

Il ne faut pas prendre ce schéma trop à la lettre. D'abord, on ne passe pas nécessairement par toutes ces étapes : beaucoup d'informaticiens de gestion diront qu'ils ne rencontrent jamais d'objets mathématiques, ce qui est d'ailleurs fort discutable. Il y a cependant souvent pour eux un intermédiaire entre les objets externes et leur représentation dans un langage de programmation, ce qu'on peut appeler des objets informatiques abstraits, plus formels que les objets externes, mais ne s'embarrassant pas de détails technologiques comme les objets d'un langage de programmation : les tables, les fichiers de l'analyse fonctionnelle en sont des exemples ; c'est sur ces objets qu'on définit les algorithmes informatiques qui traduisent le traitement. Le passage de l'objet externe à l'objet informatique abstrait, et du traitement à l'algorithme informatique est le rôle de l'analyste, celui de l'objet et de l'algorithme informatiques à l'objet du langage de programmation et au programme est le rôle du programmeur (ou de l'analyste-programmeur ; on pourrait introduire encore une étape supplémentaire, l'analyse organique, en différenciant plusieurs niveaux d'objet et d'algorithme informatiques). Même lorsqu'on passe par des objets mathématiques, il peut d'ailleurs y avoir aussi une étape de ce genre par des objets informatiques abstraits. Tout dépend du langage employé car un objet informatique (par exemple une liste) peut appartenir à un langage de programmation (langages de traitement de listes) et pas à un autre (Fortran ou PL/1).



Le passage des objets externes aux objets mathématiques est une formalisation. D'ailleurs, l'analyse est aussi en grande partie une formalisation. Et toutes les étapes des schémas précédents peuvent être considérées comme des étapes de formalisation, la "sémantique" devenant de plus en plus explicite à chaque étape.

Les êtresnommés ci-dessus, objets informatiques abstraits, objets des langages de programmation (nous avons déjà dit qu'il n'y a pas de différence essentielle entre ces deux types d'objets) et objets en mémoire seront appelés des informations (en anglais : data). Nous les étudierons ici en eux-mêmes, en faisant abstraction des applications où on les rencontre : il s'agit d'une démarche scientifique normale et c'est en ce sens que ce cours traite d'informatique théorique. Nous rappellerons aussi les objets mathématiques auxquels ils correspondent. D'autre part, nous emploierons pour les étudier les ressources des mathématiques, essentiellement le langage des ensembles.

Comme on le voit sur les schémas, il faudra savoir passer d'un type d'information à un autre, autrement dit représenter une information par une autre. Ce problème doit être résolu en particulier lorsqu'on étudie l'implémentation d'un langage, et nous nous intéresserons beaucoup à cet aspect. Mais il se pose aussi avec acuité dès qu'on traite des objets un peu complexe, par exemple dans la construction des systèmes d'exploitation, l'écriture des compilateurs, l'étude des banques d'information, les problèmes d'intelligence artificielle... D'autre part, une bonne théorie des structures d'information serait un grand pas vers l'étude de la sémantique des langages de programmation et vers l'automatisation de la construction des compilateurs.

Examinons maintenant ce que recouvre le mot d'information.

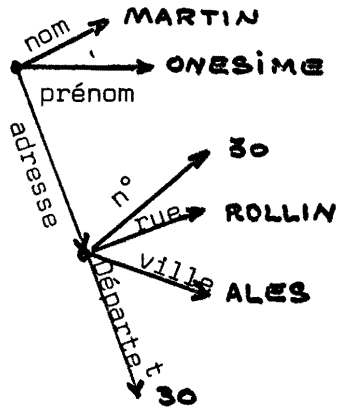
1.2. La nature des informations

Les informations peuvent être élémentaires (un entier, une valeur logique ...) ou composées à partir d'informations élémentaires. Etudions des exemples.

a) Un tableau d'objets réels, en tant qu'objet mathématique ou objet externe, est considéré comme un groupement de nombres réels. En réalité, en tant qu'objet

d'un langage ou objet en mémoire, la chose importante est la manière dont on a accès à ces nombres réels : chacun d'eux est déterminé par des indices. Un tableau de nombres réels, à deux dimensions par exemple, peut donc être considéré comme une fonction qui associe à un couple d'entiers - dans un certain domaine de définition - un nombre réel.

b) Dans un fichier d'adresses (objet externe) chaque article peut être schématisé en tant qu'objet mémoire de la manière suivante :



Nom, département, adresse ... sont des fonctions qui permettent l'accès à une chaîne de caractères, à un entier ou à un autre élément qui ne sert que de relais, de repère.

c) La table des identificateurs d'un assembleur (symboles) est encore une fonction qui fait passer d'un identificateur à l'adresse qu'il représente.

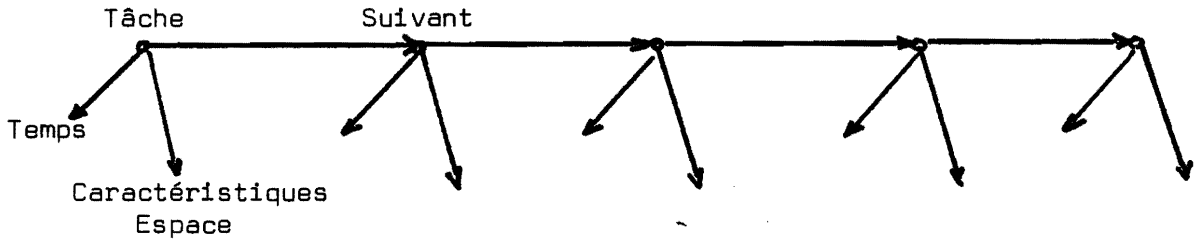
DS

DS

DS

DS

d) La file d'attente des tâches dans un système peut être schématisée ainsi :



Chaque travail, sauf le dernier, a un suivant dans la file ; d'autre part, il est caractérisé ici par deux informations élémentaires.

e) Envisageons la compilation de déclarations dans un langage évolué, par exemple Algol 68 :

réel x : = 3.14 ; compl z : = (1.2, 2.14) ; compl u : = z

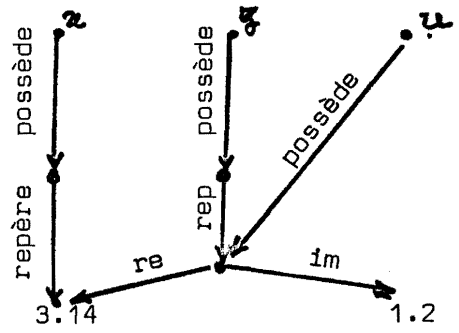
L'identificateur x est mis pour (on dit possède ou représente) un nom de réel qui à son tour donne accès à (on dit repère) la valeur 3.14 (après une nouvelle affectation, il pourra repérer une autre valeur).

De même, z possède un nom de complexe formé de deux composantes : sa partie réelle 1.2 et sa partie imaginaire 3.14.

Enfin, u possède un nombre complexe (qui ne pourra varier) : celui que repère z.

L'état de la "Machine Algol 68" après ces déclarations est schématisé sur la figure.

Possède, repère, re, im, sont ici encore des fonctions permettant des accès.



On aperçoit sur ces exemples qu'une information est formée d'objets élémentaires (valeurs élémentaires ou "relais") liés par des fonctions d'accès, souvent à une variable, mais pouvant aussi admettre plusieurs variables (exemple a).

L'opération essentielle sur ces fonctions est leur composition qui, à partir des accès élémentaires, fait passer à des accès plus élaborés.

Dans la suite, nous définirons les informations par un couple formé :

- d'un ensemble E (des objets élémentaires),
- d'un ensemble L (des accès élémentaires), ensemble de fonctions de E

Nous appelons fonction de E à p variables toute fonction à valeur dans E, définie dans une partie d'un ensemble E^p ($p \geq 0$).

Une fonction à zéro variable est une fonction de $E^0 = \emptyset$, ensemble vide, dans E.

1.3. Contenu du cours

Au cours des chapitres suivants, on étudiera divers types d'informations dans l'esprit qui vient d'être précisé, chaque type étant défini en imposant des conditions sur les ensembles E et L précédents, sans cependant chercher à établir une théorie générale.

Les informations doivent être représentées en mémoire. Pour cela il faut :

- représenter les objets élémentaires de l'information.
- représenter les fonctions d'accès par des accès à l'intérieur de la mémoire.

Nous aurons donc à préciser les accès permis en mémoire : un état de mémoire sera lui aussi considéré comme une information, avec la définition précédente ; pour permettre l'étude des représentations, il est en effet indiqué de traiter les informations " physiques " et les informations " logiques " (ou objets externes) dans le même cadre. L'étude des accès en mémoire va être faite au paragraphe 1.4 et elle nous permettra surtout de préciser nos notations. La représentantation des éléments conduit, elle, aux problèmes d'allocation de mémoire (chapitre 5).

D'autre part pour chaque type d'information, nous aurons à préciser quelles modifications sont possibles sur les informations. Dans ce cours on considère qu'une structure d'information est définie non seulement par la donnée d'un type d'information mais aussi par celle d'un ensemble de modifications élémentaires, pouvant se composer.

Nous ne traiterons que de mémoires adressables (à l'exclusion de mémoires à accès séquentiel et de mémoires associatives) et nous appellons mot l'unité d'adressage envisagée pour une application donnée sans que cette notion coincide nécessairement avec celle de mot physique : ce peut être l'octet par exemple.

Nous ne définissons pas davantage les notions de langage de programmation que nous utilisons pour décrire les algorithmes : nous supposons bien connus par exemple les notions de déclarations d'entiers et d'opérations sur les entiers, d'instruction conditionnelle, de boucle, d'itération et de procédure récursive.

Ce cours n'est qu'un effort pour séparer et mettre sous forme didactique des notions bien connues mais souvent présentées de manière disparate et fort mélangée. Il ne prétend cependant pas à une parfaite rigueur car la rigueur qui est ici possible semble conduire à des complications qui font perdre de vue l'essentiel.... signe qu'on n'est pas encore parvenu à une bonne théorie et qu'il reste du travail. C'est donc un compromis certainement discutable entre Brillat - Savarin et Bourbaki.

1.4 Définitions relatives à une mémoire adressable

1.4.1 Objets et accès élémentaires

Les objets élémentaires à considérer sont de deux sortes :

- Les mots dont l'ensemble est appelé M ; suivant les cas l'ensemble M peut être considéré comme composé d'objets physiques (emplacements), ou pour des raisons de commodité d'objets mathématiques (adresses) : ensemble des suites de p chiffres binaires, où p est un entier donné, par exemple p = 17.

- Les valeurs susceptibles d'être contenues dans les mots ; leur ensemble est appelé V. Ce peut-être par exemple l'ensemble des suites de 32 chiffres binaires.

Les fonctions d'accès élémentaires sont :

a) La fonction contenu notée c, qui est une application de M dans V : à tout mot m, elle associe la valeur de son contenu.

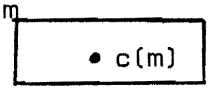
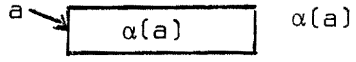


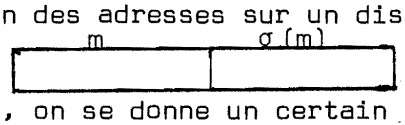
schéma de la fonction C.

b) La fonction d'adressage, notée α : certaines valeurs de l'ensemble V (par exemple celles dont les 15 premiers chiffres binaires sont des zéros) peuvent être considérées comme des " adresses " , donnant accès à des mots. α est une fonction de V dans M dont l'ensemble de définition est une partie A de V ; α est supposée être injective.

Par exemple si M est l'ensemble des suites de 17 chiffres binaires, V celui des suites de 32 chiffres binaires, α peut être la fonction enlevant les 15 bits de poids forts, lorsqu'ils sont nuls.



c) Une fonction de succession faisant passer d'un mot au suivant dans la mémoire physique. Il s'agit d'une fonction σ de M dans M, dont la fonction réciproque est notée σ^{-1}.
Il s'agit souvent de la fonction qui fait passer d'un mot d'adresse entière a, au mot d'adresse a+1, mais il peut s'agir d'une fonction plus complexe, par exemple pour la gestion des adresses sur un disque.



d) De plus dans chaque cas, on se donne un certain nombre de fonctions de V. (au sens défini à la fin de 1.2) : elles traduisent les calculs élémentaires sur les valeurs de V. Nous appelons calculs les fonctions obtenues en composant les fonctions d'accès élémentaires de ce type.
Une fonction de ce type est la fonction (sc) qui associe à chaque valeur de V un " successeur ", sc(v). Par commodité sc(v) sera le plus souvent noté v+1 et $sc^k(v)$ noté v+k : ceci ne suppose cependant aucune relation arithmétique entre v et v+1. Cette fonction est définie par rapport à la fonction σ de telle sorte que

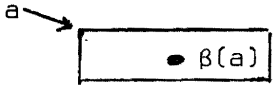
$$\alpha.sc(u) = a(u+1) = 6.\alpha(u)$$

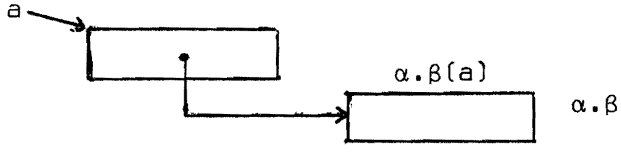
e) Si on n'a pas directement accès aux mots, on a accès aux valeurs de V ; autrement dit, parmi les fonctions d'accès élémentaires se trouvent les éléments de V, fonctions à 0 variable.
C'est par exemple l'élément u dans l'exemple e) du paragraphe 1.2.

1.4.2 Composition des accès élémentaires.

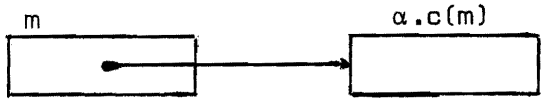
Les accès élémentaires peuvent être composés, nous venons de le voir dans les exemples précédents. Etudions qùelques fonctions particulières :

- $\beta = \subset \bullet \alpha$ est une fonction de V dans V, définie dans l'ensemble A $\subset$ V de définition de α. Elle fait passer d'une adresse au contenu du mot admettant cette adresse, elle réalise donc l'adressage direct. De la même manière la fonction $\alpha \bullet \beta$ réalise l'adressage indirect.

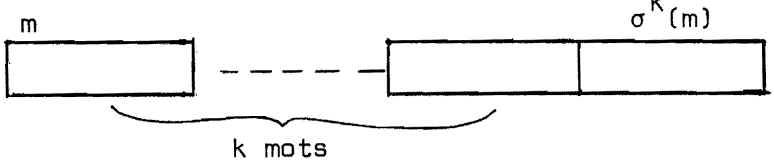




- $\alpha \bullet c$ est une fonction de M dans M qui fait passer d'un mot m au mot dont l'adresse se trouve dans m ; elle s'appelle chaînage. Plus généralement si f est une fonction de V dans V, $\alpha \bullet f \bullet c$ est encore appelé chaînage.



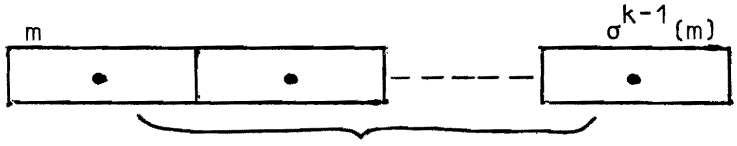
- $\sigma \bullet \sigma = \sigma^2$ et plus généralement σ^k sont des fonctions de M dans M (succession de mots de K en K)



1. 4. 3. Passage d'un mot à un k-uple de valeurs

On a souvent vesoin de passer d'un mot non à une valeur, mais à un k-uple de valeurs, autrement dit de définir une application d'une partie M' de M dans V^k.

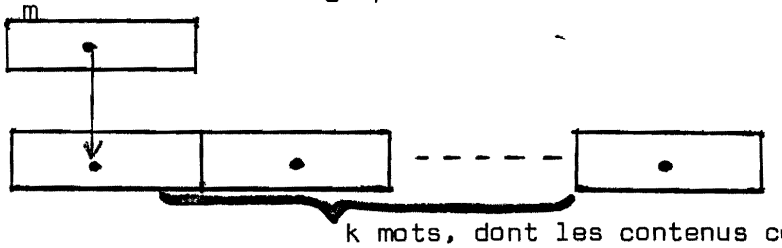
a) Le plus simple est l'accès direct par contiguité qui généralise la fonction c ; nous appelons cette fonction c_k :



k mots, dont les contenus constituent $C_k(m)$

$$C_k = (C, C.\sigma, \ldots C.\sigma^{k-1})$$

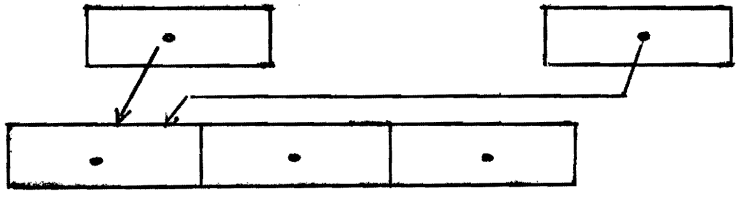
b) Accès indirect : chaînage préalable



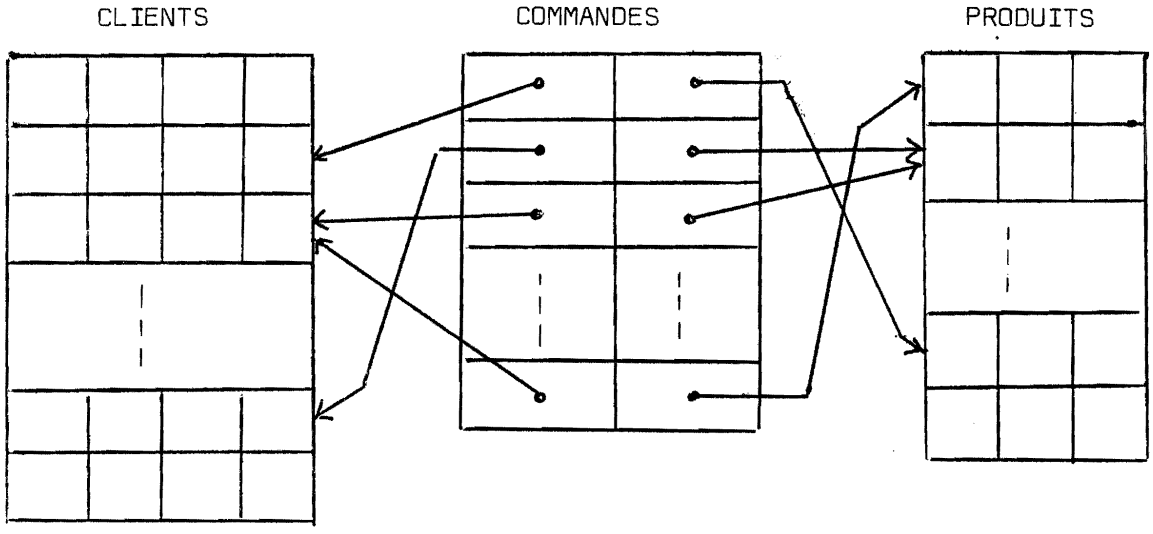
$C'_k = C_k.\alpha.C$

k mots, dont les contenus constituent $C'_k(m)$

L'accès indirect permet un " partage de valeurs " quand plusieurs mots conduisent à la même valeur.



D'où un gain de place, surtout lorsque k est grand.
De plus, une modification de la fonction c'_k en un mot demande la modification de ce seul mot si le k-uple à repérer existe déjà en mémoire.
Exemple : Gestion d'un carnet de commande : chaque commande est caractérisée par des renseignements sur le client et sur le produit commandé ; on la représente par deux mots qui donnent accès à ces renseignements dans deux autres tables.



L'accès indirect présente un intérêt supplémentaire lorsqu'il n'y a pas assez de place entre les mots de l'ensemble M' où est définie la fonction d'accès pour permettre d'y ranger les k-uples de valeurs.
Ceci se présente surtout quand k peut varier au cours du traitement (application de M' dans $V^* = \bigcup_k V^k$.

Les valeurs peuvent d'ailleurs se trouver en mémoire secondaire alors que les mots y donnant accès par chaînage sont en mémoire centrale (dictionnaire de mots clés, tables de fonctions standards, etc.).

1. 4. 4. Modification de l'état d'une mémoire adressable :

Les informations qui viennent d'être étudiées sont les états d'une mémoire adressable. La seule modification élémentaire qu'on peut leur faire subir est le changement de la fonction c sur un mot ; les autres fonctions

d'accès ne peuvent être modifiées. Si $m \in M$ et $v \in V$, le changement de c(m) pour lui donner la valeur v est noté :

$$m := v \quad \text{ou} \quad v \rightarrow m$$

Par exemple, a et a' étant deux adresses, $\beta(a') \rightarrow \alpha(a)$ transfère dans le mot d'adresse a le contenu du mot d'adresse a'. On a alors $\beta(a) = \beta(a')$.

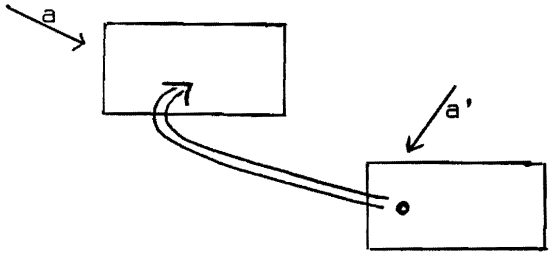


Schéma d'une modification du contenu d'un mot par le contenu d'un autre.