CHAPITRE II

Bâtons, chiffres et lettres

Un événement extérieur : voilà ce qui, en général, déclenche le fonctionnement d'une intelligence ou d'un automate. Encore faut-il que cet événement se manifeste, *se signale,* de quelque façon. Ce signal peut être très élémentaire, et n'apporter d'autre renseignement que d'apprendre qu'un événement s'est produit (bruit qui éveille l'attention, mise sous tension d'un appareil, etc.). S'il est plus structuré, il peut fournir une information relativement plus détaillée (bruit d'une assiette qui se brise, aboiement d'un chien).

Le *signal, plus* ou moins riche en *informations,* est donc ce qui permet la *communication* entre le milieu extérieur et l'intelligence en action (ou l'automate). Bien entendu, les phénomènes naturels les plus divers peuvent lui servir de support. La seule contrainte qu'on doive leur imposer est la simple possibilité d'avoir une action sur l'être intelligent ou sur l'automate.

La vision d'un arc-en-ciel, la perception d'une odeur de caoutchouc brûlé, la réception par la voie du courrier d'un prospectus publicitaire, voici autant de messages que, dans un contexte déterminé, il sera facile de déchiffrer. L'information qu'ils transmettent est purement et pauvrement qualitative.

D'autres messages sont plus explicites et expriment d'une certaine façon la *structure* de l'événement qu'ils s'efforcent de signaler : le spectre optique d'un échantillon donne une indication sur sa composition chimique, le diagramme de diffraction d'un cristal est lié à sa structure géométrique. De même la *photographie* d'une maison rend-

elle manifestes certaines des propriétés géométriques (et quelques autres) de la maison elle-même. Par contre le *plan* de la maison ne retient plus que certains aspects de la géométrie et y ajoute des informations d'un type différent comme les cotes. Enfin une *description dans un langage particulier,* par exemple le français moderne, de cette maison nous éloigne encore plus de l'objet initial.

Ce cas particulier nous permet d'apercevoir, en raccourci, le chemin parcouru depuis le phénomène de la *signalisation* jusqu'à celui de la *description codée.* Dans les premières étapes les rapports de structure entre l'événement et son signal demeurent transparents : l'image est encore une *analogie.* A la fin de cette chaîne de substitutions l'image n'est plus qu'un *symbole* ou, mieux, qu'un agencement de symboles réglé selon un jeu de conventions données d'avance. Bien entendu la perte dans le pouvoir expressif du signal est compensée par un gain dans la maniabilité, et c'est là sans aucun doute la raison fondamentale du développement et du succès des systèmes linguistiques, naturels ou artificiels.

Même si l'on se restreint aux messages qui sont recevables par nos organes des sens (ou par ceux de nos automates), on remarquera que la nature et la structure de ceux-ci sont fortement conditionnées par le nombre de dimensions (au sens géométrique usuel) qui leur est accordé.

Les deux types principaux de messages sont, on le sait, ceux du type graphique bidimensionnel et ceux du type sonore unidimensionnel. Ce qui les distingue essentiellement, c'est évidemment la possibilité de disposer les signes qui le constituent selon un arrangement autre que l'arrangement linéaire. Cette possibilité est extrêmement intéressante comme le montre l'exemple suivant.

Considérons le dessin de la figure 3. Tout automobiliste placé devant un panneau qui contient ce dessin l'interprétera aisément comme un ordre indiquant que la première route à droite après le pont est en sens interdit.

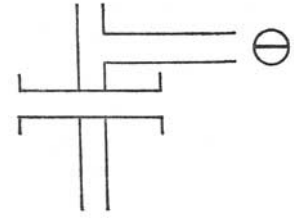


FIG. 3. - Exemple de message bidimensionnel

Il y a donc là une différence fondamentale dans les conditions d'exploitation des

deux types de messages et nous aurons l'occasion à plusieurs reprises de souligner certaines conséquences de cet état de fait. Par contre il existe une ressemblance dans les conditions d'exploitation qui est l'identification de deux signaux qui ne diffèrent que par des détails considérés comme non significatifs. C'est ainsi que le dessin de la figure 3 apporte un message qui sera identifié à celui du panneau de signalisation routière lui-même, malgré la différence considérable des dimensions, de la couleur, etc.
De même le message sonore en morse sera accepté par un auditeur compétent, qu'il soit émis par un sifflet ou un tambour.

SIGNAL ET CODIFICATION

Il faut donc examiner un peu en détail le problème de la codification. Il est clair que, grâce à la codification, on peut associer à des événements extérieurs les deux types de signaux qui les représentent. Les uns sont eux-mêmes des événements qui développent une certaine ressemblance avec ce qu'ils représentent, c'est ce que l'on appelle le codage *analogique.* Les autres sont des séquences d'objets que l'on utilise en tant que réalisations particulières du concept de nombre. C'est ce qu'on appelle le codage *digital* (ou arithmétique).
Au premier type appartient par exemple le tachymètre qui associe à la vitesse de déplacement d'un véhicule la position d'une aiguille devant un cadran gradué. Au second appartient la balance automatique qui imprime le poids, en chiffres et lettres, sur un ticket de carton.
Une grande variété de systèmes de codage-décodage existe dans la nature aussi bien que dans les machines. Mais la structure de certains de nos organes sensoriels (l'ouïe notamment) les contraint à n'accepter les signaux que *séquentiellement.* Et notre langage humain, après être passé par un stade purement oral, a vu même sa forme écrite se plier à cette exigence de séquentialité.
Aussi, pour étudier la reconnaissance d'un signal type isolé au milieu d'informations légèrement différentes, est-il possible de se limiter au cas d'une structure unidimensionnelle.
Dans le cas général, ce signal peut être représenté par un graphique qui décrit l'évolution d'une certaine grandeur en fonction du temps.

Si l'on observe le système nerveux d'un être vivant cette grandeur sera souvent une tension ou un courant électrique. Il en sera de même pour la plupart des automates (fig. 4).

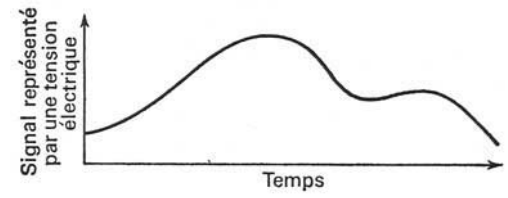


FIG. 4. - Représentation graphique d'un signal analogique élémentaire

Lorsqu'on veut se servir d'un signal analogique comme information quantitative, il est nécessaire que la précision de la lecture ne soit pas affectée par les perturbations extérieures au phénomène représenté.

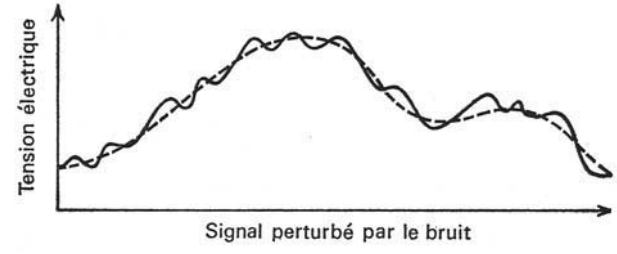
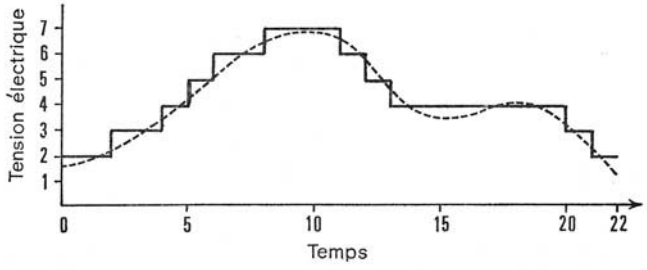


FIG. 5. - Echantillon d'un signal
qui sera identifié au signal indiqué en pointillé

La reproduction du signal comme sa transmission sont souvent accompagnées de l'introduction d'éléments parasites, que, dans le cas de l'électronique, et dans des domaines plus vastes, on appelle souvent le « bruit de fond». C'est ainsi que les figures 4 et 5 représenteront le même événement.
Pourquoi une « réduction » du type précédent est-elle nécessaire ? C'est évidemment pour limiter à un nombre fini de possibilités l'infinie

variété des événements et des objets. Mais alors on aperçoit une procédure particulièrement simple pour opérer une telle réduction : c'est la *quantification.* Au lieu de considérer que le temps et la tension électrique peuvent varier de façon continue, on ne s'intéresse qu'aux valeurs prises par la tension en une suite régulière d'instants. De même on ne distingue pas les valeurs de la tension dont la différence est inférieure à une certaine quantité.

Ce processus de quantification est explicité dans la figure 6 : le signal quantifié représenté en trait plein est, comme celui de la figure 5, assimilable à celui de la figure 4. Mais le « bruit » qui l'en distingue est cette fois un bruit artificiel : le « bruit de quantification ».



Temps	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Tension	2	2	3	3	4	5	6	6	7	7	7	6	5	4	4	4	4	4	4	3	2

FIG. 6. - Le signal quantifié
et sa réduction en une fonction numérique

Le signal est ainsi réduit à une simple suite de symboles (car on peut supprimer la première ligne du tableau de la figure 6, qui ne fait qu'énumérer les symboles successifs). On est ainsi passé du signal « analogique» à un signal « digital » ou « arithmétique ». Mais ici ne s'arrête pas le travail de *codification* car la transmission du signal digital peut elle-même comporter l'apparition de nouvelles erreurs.

Un codage supplémentaire peut dans une large mesure conjurer ce défaut, comme on peut le voir dans l'exemple suivant

Soit un message constitué par le signal digital *binaire* (c'est-à-dire construit à l'aide de deux symboles seulement) suivant

I O I I O I.

Substituons à ce signal un signal obtenu en répétant trois fois chaque symbole. Nous obtenons

III OOO III III OOO III.

Supposons maintenant que des erreurs se glissent lors de la transmission et que le signal reçu soit

OII OIO III IOI OOI OII.

Si le destinataire *décode* le message en remplaçant chaque suite de trois symboles par celui qui s'y trouve en majorité, il obtiendra un signal identique au signal de départ (le lecteur le vérifiera sans peine) et ceci malgré cinq erreurs qui se seront produites pendant la transmission (I).

Ceci montre que les garanties dont on peut s'entourer pour éliminer l'effet des sources d'erreur a pour effet d'alourdir le travail de codage et - dans le cas de codes séquentiels - d'allonger la représentation symbolique des signaux.

Il se pose donc un problème de recherche de codes optimaux, tant par l'efficacité de leur élimination des erreurs que par la simplicité de leur mise en oeuvre dans la phase de codage comme dans la phase de décodage.

Dans ce qui suit, nous n'entrerons pas dans les détails des problèmes théoriques et pratiques que pose le problème du choix d'un codage optimal. Il suffit de préciser ici que nous nous bornerons désormais au codage digital dans le cas d'événements ou d'informations discontinus, de telle sorte que l'erreur de quantification est absente.

(i) Cet exemple est emprunté à D. SLEPIAN dans son article Coding theory paru dans le numéro spécial de *Nuovo Cimente* consacré à la théorie de l'information (1959), vol. 13, p. 373.

QUELQUES EXEMPLES DE CODIFICATION DIGITALE

Le titre même du chapitre en cours : « Bâtons, chiffres et lettres », titre que nous avons emprunté (avec son accord) à Raymond Queneau, met l'accent sur ce choix que nous effectuons ici de codes digitaux pour le traitement des problèmes de l'intelligence artificielle. Il n'implique nullement, il convient de le souligner, que des codes analogiques sont inutilisables ici. Au contraire, il semble bien que le cerveau, lui, utilise des techniques de codification hybrides, c'est-à-dire combinant les méthodes analogiques et digitales. Ce point a été souligné par J. von Neumann (1). Nous avons nous-même observé à plusieurs reprises (pour des problèmes de simulation de jeux et pour des problèmes de documentation automatique) que des techniques analogiques se présentaient assez naturellement dans certains cas.

La restriction que nous nous imposons n'est donc motivée que par la nécessité de nous limiter. Elle illustre aussi notre parti pris de ne tenter aucun rapprochement entre la technologie de l'intelligence artificielle et le fonctionnement du système nerveux central. Encore une fois - contrairement par exemple à ce que l'on trouve dans la littérature soviétique - nous séparons nettement intelligence artificielle et cybernétique.

La codification digitale est essentiellement une manipulation de symboles. En première approximation c'est même la forme la plus simple que l'on puisse concevoir pour une telle manipulation : le message à coder contient une suite de symboles, le message codé est obtenu par une substitution un à un des symboles, en se référant à une table d'équivalences.

La figure 7 donne un exemple de table d'équivalences multiples permettant la traduction d'un message rédigé en lettres usuelles (majuscules), dans le code télégraphique, ou dans un code pour machine à calculer.

La table ci-contre fait apparaître un phénomène intéressant : la première colonne propose en effet une suite de symboles qui sont tous différents : ce sont des *lettres* d'un *alphabet* et elles sont nécessairement distinctes.

(1) Notamment dans son ouvrage The computer and the brain cité en bibliographie.

Alphabet romain	Morse	Code octal BCD	Code Hollerith
A	·−	BA 1	12-1
B	−···	BA 2	12-2
C	−·−·	CBA 21	12-3
D	−··	BA 4	12-4
E	·	CBA 4 1	12-5
F	··−·	CBA 42	12-6
G	−−·	BA 421	12-7
H	····	BA8	12-8
I	··	CBA8 1	12-9
J	·−−−	CB 1	11-1
K	−·−	CB 2	11-2
L	·−··	B 21	11-3
M	−−	CB 4	11-4
N	−·	B 4 1	11-5
O	−−−	B 42	11-6
P	·−−·	CB 421	11-7
Q	−−·−	CB 8	11-8
R	·−·	B 8 1	11-9
S	···	C A 2	0-2
T	−	A 21	0-3
U	··−	C A 4	0-4
V	···−	A 4 1	0-5
W	·−−	A 42	0-6
X	−··−	C A 421	0-7
Y	−·−−	C A8	0-8
Z	−−··	A8 1	0-9

FIG. 7. — Codifications de l'alphabet usuel

Au contraire les colonnes suivantes proposent des symboles qui sont décomposables chacun en une *suite* de symboles plus élémentaires. On peut considérer chacun d'eux comme un *mot* construit sur un alphabet primitif. Pour la deuxième colonne il s'agit de l'alphabet réduit aux signes . et -. Pour la troisième colonne, cet alphabet est formé des signes C, B, A, 8, 4, 2 et I. Enfin, pour la dernière colonne, il s'agit des signes 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11 et 12 et du signe -.

On remarquera aussi que, dans le cas de la deuxième colonne, les symboles sont, par rapport à leur « alphabet interne », des mots de

longueur inférieure ou égale à quatre lettres. Dans le cas de la troisième colonne les « mots » sont de longueur *impaire* (mais inférieure ou égale à cinq). Enfin la colonne correspondant aux code Hollerith a aussi une structure très particulière (tiret entre deux chiffres dont le premier est nécessairement **I2,** II ou 0).

Lorsqu'on veut procéder au codage ou au décodage, le passage d'un signe à celui qui lui correspond dans un autre code peut être réalisé par un automate très simple : un premier organe *lit* le symbole d'entrée, un second *consulte une table* d'équivalence et choisit le couple dont le premier élément est le symbole d'entrée, un dernier organe *écrit* le second élément du couple ainsi détecté.

C'est ainsi que pour trouver l'équivalent Hollerith de la lettre « N » on consulte la table formée de la première et de la dernière colonne de la figure 4. On détectera le couple (N, II-5) et on donnera le résultat « **II-5** ».

On pourrait aussi songer à utiliser la structure particulière qu'offrent les codes des colonnes 2 à 4 afin de diminuer non pas la complexité de l'automate, mais plutôt l'encombrement de sa mémoire. Nous aurons d'ailleurs l'occasion de revenir sur cette possibilité.

Notons en tout cas que la structure du code traduit souvent les propriétés et les contraintes du matériau qui sert de support au message. C'est le cas des codes qui correspondent aux colonnes 3 et 4. En effet, en ce qui concerne la colonne 3, les 6 symboles I, 2, 4, A, B, C peuvent être mis en correspondance avec des perforations dans une bande de papier ou avec des traces d'une bande magnétique. Dans le cas de la colonne 4, les chiffres de 0 à 12 peuvent être mis en correspondance avec les lignes d'une carte perforée.

Dans le cas 3, les lettres de l'alphabet sont donc associées à des ensembles de 3 ou 5 perforations de la bande; dans le cas 4, elles sont associées à des ensembles de 2 perforations de la carte dans une même colonne.

DES CODES AUX LANGAGES

En examinant la structure de certains codes élémentaires que nous avons présentés dans le tableau de la figure 7, nous avons utilisé deux expressions qui amorcent un long développement : ce sont «alphabet » et « mot construit sur un alphabet ».

Si le but de la mise sous forme symbolique d'une information était simplement le remplacement d'un signal par une suite convenablement choisie de signes, les problèmes seraient relativement élémentaires. L'exemple du paragraphe précédent nous montre que des *mots* construits à l'aide d'un alphabet élémentaire (par exemple les brèves et les longues de l'alphabet morse) peuvent servir à coder les *lettres* d'un alphabet plus grand (ici l'alphabet ordinaire) dont les mots seront à leur tour les lettres d'un super-alphabet (le lexique), etc.

Codage et décodage sont des manipulations de symboles au niveau de l'orthographe qui n'est que le niveau de base dans la hiérarchie des moyens d'expression formalisés.

Or, au cours du chapitre I nous avons observé que la mise en oeuvre de l'intelligence se fait au moyen de substitutions dont le codage est un exemple simple, mais de substitutions enchaînées selon les divers niveaux d'une hiérarchie complexe.

C'est ce qui nous conduit, à peine confrontés aux problèmes de codage, à aborder les questions du langage, tout au moins sous quelques aspects élémentaires qui commandent tous les autres.

C'est ainsi qu'au-dessus du niveau alphabétique, nous distinguerons, suivant en cela une nomenclature traditionnelle, trois niveaux dans l'activité de communication

- un niveau *syntaxique* qui détermine l'adéquation *interne* des expressions qui constituent le message ;
- un niveau *sémantique* qui détermine l'adéquation de l'expression linguistique *avec l'événement* ou le concept qu'elle entend communiquer ;
- un niveau *pragmatique* qui détermine l'adéquation de l'action déclenchée par le message avec le contenu de celui-ci.

Bien entendu, la distinction de ces niveaux devient, dans certains cas limites, assez arbitraire, mais nous n'aurons pas à nous en préoccuper ici (I).

(I) Cf. l'article de Léo APOSTEL dans le volume *Logique et connaissance scientifique,* publié sous la direction de J. PIAGET dans *l'Encyclopédie de la Pléiade.*

Chaque niveau est caractérisé par la mise en oeuvre de *contraintes* règles, limitations de toutes sortes (forme ou longueur des mots comme dans l'exemple de la figure 7, leur disposition et leur enchaînement, leur répartition en catégories ordonnées et ainsi de suite). Ces contraintes ont évidemment pour conséquence de réduire le nombre de combinaisons acceptables et ceci a un effet correcteur d'erreurs comme dans l'exemple de la page 27. Cela a aussi une signification plus profonde, *anticombinatoire,* que nous ne voulons que mentionner ici, afin d'y revenir plus à loisir à la fin de l'ouvrage.

Les diverses contraintes se distinguent aussi par leur *rayon d'action.* Les contraintes alphabétiques sont évidemment purement locales elles précisent les possibilités de choix des mots eux-mêmes. Les contraintes syntaxiques mettent en cause les mots voisins. Les contraintes sémantiques établissent des liaisons au sein d'un texte tout entier et même, au-delà du texte, se réfèrent à toute une culture, une histoire, une tradition. Les contraintes pragmatiques enfin nous font sortir du domaine propre du langage et mettent en action les locuteurs eux-mêmes. Il est important de garder présentes à l'esprit ces diverses remarques lorsqu'on aborde le domaine des automates, car elles sont utiles dans ce domaine également.

Dans le langage mathématique, on s'en doute, l'aspect pragmatique est délibérément négligé. Par contre l'aspect sémantique, quoique souvent caché, n'est jamais complètement absent : il doit y avoir une correspondance entre les symboles numériques et les nombres eux-mêmes, entre les symboles géométriques et les figures, etc.

Par contre la logique mathématique, et plus précisément la théorie des systèmes formels, s'intéresse exclusivement à l'aspect syntaxique.

Le langage naturel, dans ses diverses réalisations particulières comme le français que nous parlons, fruit d'une longue évolution, soumis à des influences multiples et souvent contradictoires, propre à décrire les événements les plus variés, à traduire les pensées les plus subtiles, s'il est un outil merveilleux pour le fonctionnement de l'intelligence naturelle, demeure très inférieur aux langages artificiels lorsqu'on veut illustrer les problèmes fondamentaux des langages. Aussi nous adresserons-nous maintenant à des langages très particuliers : les systèmes formels.

DES LANGAGES AUX SYSTEMES FORMELS

La méthode normale, pour définir un système formel, consiste en effet à se donner un *alphabet* de signes, puis à sélectionner, parmi les combinaisons de ces signes, celles qui seront admises et qu'on appellera les *expressions bien formées,* et enfin, parmi les suites possibles d'expressions bien formées, on distinguera un certain nombre de *schémas déductifs* pour lesquels la dernière expression de la suite sera considérée comme la conséquence de l'ensemble de celles qui la précèdent.

Pour fixer les idées, supposons que l'alphabet soit composé des 12 signes

0, I, 2, 3, 4, 5, 6, 7, 8, 9, +, =

(les I0 premiers sont appelés chiffres).

Les expressions bien formées sont définies de la façon suivante :

Si x est un chiffre, x est un entier.

Si x est un chiffre et y un entier,
xy et yx sont des entiers.

Si $y_1 y_2 y_3$ sont des entiers,
$y_1 + y_2 = y_3$ est une expression bien formée.

Les schémas déductifs sont simplement

(0 + 0 = 0)
(0 + I = I)
(0 + I = 2) (0 + 2 = 2)
(0 + I = 3) (I + 2 = 3)
....
(8 + I = 9) (7 + 2 = 9) ..... (0 + 9 = 9).

Il y a visiblement 55 schémas de cette sorte que nous dénoterons par S_{ij} où i et j vont de 0 à 9, mais $i + j < 10$.

Le dernier schéma est alors, si y et z sont des entiers de la forme $y_1 x_1$ et $z_2 x_2$, x_1 et x_2 étant des chiffres

$$S_R(y, z) : \quad (y_1 x_1 + z_2 x_2 = S_R(y_1, z_2)\, S_{x_1 x_2}) \quad \text{(I)}.$$

(I) Par abus de langage, S_{ij} désigne aussi le résultat de l'opération qu'il dénote.

Nous avons ainsi une formalisation de l'addition des entiers dans la représentation décimale, dans le cas où l'addition peut s'effectuer sans report.

Malgré son caractère élémentaire, l'exemple ci-dessus permet de mettre en évidence plusieurs notions intéressantes.

Tout d'abord, au niveau du système formel proprement dit, on remarquera que la distinction, au sein de l'alphabet, des chiffres et des autres signes, permet d'engendrer une représentation des expressions sous forme de graphe, c'est-à-dire une représentation à deux dimensions ; c'est ainsi qu'à une expression telle que

$$2 + 3 = 5$$

on est conduit naturellement à associer une représentation graphique qui est celle de la figure 8.

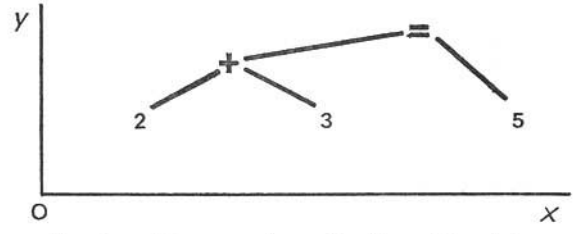


FIG. 8. — Une expression arithmétique élémentaire représentée à l'aide d'un « graphe »

En projection sur l'axe des x, on obtient bien l'expression initiale, mais en projection sur l'axe des y on obtient la forme

$$= + 2\ 3\ 5$$

qui n'est autre que la fameuse notation « sans parenthèses » de Lukaciewicz (souvent appelée notation polonaise).

Il sera souvent commode d'utiliser des représentations bidimensionnelles de ce type (représentations « sagittales ») au lieu de la représentation séquentielle usuelle nécessaire pour l'introduction dans les automates.

Nous tenons à souligner la possibilité qu'offrent les règles syntaxiques (ici la disposition des signes d'opération et des symboles arithmétique) pour réduire sous forme d'un mot, c'est-à-dire d'une suite linéaire de symboles, un système de rapports qui s'exprimerait plus naturellement dans une représentation à plusieurs (en tout cas à deux) dimensions.

Une seconde remarque qui se dégage de notre exemple est liée au schéma S_R qui en réalité n'est pas un *schéma* mais une *famille de schémas* dépendant de deux paramètres.

Nous verrons plus loin la portée de cette remarque.

Mais dès maintenant nous voulons souligner que le modèle d'un système formel défini sur un alphabet et soumis à diverses contraintes de type morphologique ou syntaxique et pourvu de règles de transformation qui sont en fait des substitutions de lettres ou de groupes de lettres recouvre pratiquement tout le domaine de l'activité intellectuelle formalisée : mathématique, logique, etc.

QUELQUES EXEMPLES DE SYSTEMES FORMELS

L'exemple développé dans le paragraphe précédent était complètement artificiel : aucune intelligence humaine, aucun automate n'utilisent un tel système formel. Mais il peut être considéré comme une bonne introduction à l'étude de systèmes plus réalistes. Nous allons les aborder dans un ordre de complexité croissante qui n'est pas directement lié à leur mise en évidence historique.

C'est ainsi que le *calcul des propositions* se constitue en un système formel dont l'alphabet est formé de symboles tels que

- A, B, C, etc., qui représentent des propositions logiques ;
- les symboles &, v, -> et ~ (qui correspondent aux expressions « et », « ou », « entraîne », et « non ») ;
- enfin les parenthèses « (», et «) ».

Les *expressions bien formées (ou* formulées) sont telles que

- A est une formule ;
- si $\mathscr{A}$ et $\mathscr{B}$ sont des formules, $(\mathscr{A}\ \&\ \mathscr{B})$ $(\mathscr{A}\ \mathrm{v}\ \mathscr{A})$, $(\mathscr{A}\ \text{-->}\ \mathscr{A})$ et $\sim\mathscr{A}$ sont des formules.

Les *axiomes* sont des formules convenablement choisies telles que

A --> (B--> A)
(A --> (B-->C)) --> ((A --> B) --> (A --> C))
etc.

Enfin les schémas déductifs comprennent des séquences telles que

$$\{\mathscr{A}, \mathscr{A} \text{ --> } \mathscr{B} \cdot \mathscr{B}\}$$

(appelées règles de détachement, ou de conclusion, ou du *modus ponens).*

Au fond le calcul des propositions est directement issu d'un fragment (relativement pauvre) du langage ordinaire. La syntaxe fait essentiellement usage des conjonctions « et » et « ou ». La sémantique correspond à une suite de jugements de valeur portés sur des événements ou des opinions, jugements de la forme : « ceci est vrai », « cela est faux ». Partant de ce calcul élémentaire et enrichissant un peu notre langage, nous obtenons le calcul des prédicats.

Le *calcul des prédicats* se constitue, lui aussi, en un système formel (qui englobe le calcul des propositions).

Ici les propositions ne sont pas des lettres de l'alphabet mais des mots.

Les lettres A, B, C, etc., dénotent des *prédicats ;* les lettres *a, b,* c, etc., sont des objets susceptibles de posséder les propriétés correspondant à ces prédicats et les propositions deviennent des expressions bien formées dans l'alphabet

A, B, C, ..., a, b, c, ..., (,)

de la forme

A (a, b, c)

De plus les signes, ∃, ∀, L, apparaissent dans des expressions bien formées telles que

(∃x) A (x)

qui signifie « il existe un objet x qui possède la propriété A ».

Ici encore, un choix convenable d'axiomes et de schémas déductifs permet de construire le système formel rigoureusement. Nous renvoyons le lecteur intéressé aux traités modernes de logique mathématique.

L'enrichissement sémantique que permet le calcul des prédicats ne semble pas considérable (l'être que, tous les êtres qui, ...). Il n'en est que plus remarquable de constater la différence essentielle qui existe dans la « richesse » des deux systèmes. Mais il est temps d'aborder le domaine du langage mathématique proprement dit.

On sait que les mathématiques peuvent se bâtir formellement à partir de la théorie des ensembles (ainsi qu'en témoigne le célèbre ouvrage de N. Bourbaki). Et la théorie des ensembles (sous certaines conditions restrictives) s'inscrit dans le cadre des systèmes formels que nous avons décrits.

On y rencontre des prédicats binaires tels que

$$\in (x, E)$$

exprimant que l'élément x appartient à l'ensemble E, ce qu'on écrira en général $x \in E$.

De même les prédicats ternaires

$$\cap(A, B, C) \quad \text{et} \quad \cup(A', B', C')$$

exprimant que A (respectivement A') est la réunion (resp. l'intersection) des ensembles B et C (resp. B' et C'), s'écrivent

$$A = B \cap C \qquad A' = B \cap C$$

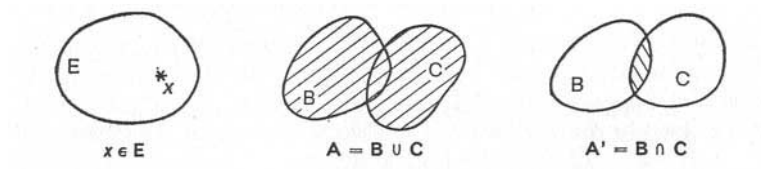


FIG 9 - Sémantique élémentaire de la théorie des ensembles

Ici encore, les diverses écritures possibles se combinent dans une représentation bidimensionnelle, représentation qui met en évidence le caractère syntaxique des contraintes du système formel.

A titre d'exemple, nous présentons ci-après la représentation d'une expression contenant des notions logiques et arithmétiques tout à la fois. Cet exemple est dû au logicien polonais R. Suszko (1).

(1) Cf. Acta Logica.

Considérons l'expression

$$\underset{x}{\mathrm{L}}\,(x \in \underline{\underline{\mathrm{N}}} \wedge \bigwedge_{y}(y \in \underline{\underline{\mathrm{N}}} \rightarrow y + x = y))$$

Cette expression définit le nombre entier x tel que, pour tout entier y on ait y + x = y (il s'agit donc en fait d'une définition du zéro de l'arithmétique).

Cette expression peut être représentée par le graphe de la figure 10

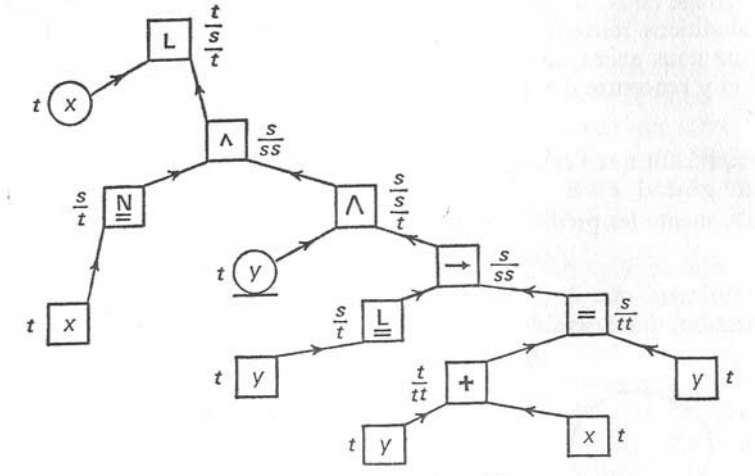


FIG. 10. — Représentation graphique d'une expression contenant des opérations et des prédicats arithmétiques et logiques

Les fractions symboliques construites à l'aide de *s et t* utilisées par Suszko expriment la variété des fonctions syntaxiques que possèdent les relations proportionnelles et les prédicats présents dans l'énoncé.

Nous nous contenterons ici de souligner l'intérêt de cette représentation dont nous retrouverons l'écho dans le chapitre V, à l'occasion de l'analyse grammaticale.

Lorsqu'on regarde de près le fonctionnement des systèmes formels, notamment aux niveaux alphabétique et syntaxique, on ne rencontre que des règles de substitutions et d'arrangement, complètement définies et automatiquement applicables.

Bien entendu le caractère « mécanique » des manipulations symboliques n'apparaît clairement que lorsque les systèmes qui les mettent en oeuvre sont effectivement formalisés ce qui, en mathématique, est une acquisition récente, et en linguistique n'est même pas encore réalisé.

Il est donc naturel que l'idée d'utiliser des « machines abstraites » pour représenter les systèmes formels soit apparue assez tardivement. Nous examinerons cette évolution avec quelques détails au début du chapitre IV.

Nous nous contenterons ici d'utiliser un modèle de machine abstraite particulièrement simple (i), celui de A. Turing (modèle introduit en 1936, donc bien avant le développement des calculatrices) comme étape intermédiaire entre le domaine des codes, langages, systèmes formels, etc., et celui des automates.

Ce faisant nous aurons, en passant par la logique et les systèmes formels (notamment mathématiques), montré l'existence d'une continuité entre le domaine de la construction symbolique qu'est le langage naturel et celui des automates, donnant ainsi déjà une certaine crédibilité à la notion même d'intelligence artificielle.

LES AUTOMATES ET LEUR LANGAGE

Rappelons, en commençant ce paragraphe, que nous nous limitons à un automate particulier, d'ailleurs abstrait, mais que cela ne comporte aucune restriction de fait, car la machine de Turing (voir le schéma de la figure ii) pourrait être effectivement construite ou plus simplement simulée sur une machine à calculer réelle (2).

Elle comporte donc une *bande* (pour Turing il s'agissait d'une bande de papier ordinaire, mais elle pourrait être tout support *séquentiel* d'information). Cette bande comporte des cases dans lesquelles une information peut être inscrite ou effacée. Cette bande peut se déplacer dans un sens ou dans l'autre par rapport à l'organe de lecture-écriture. Elle comporte aussi un organe susceptible de se trouver dans

(r) Simple du point de vue didactique, si l'on veut introduire les notions de sous-programme, de bloc-diagramme, etc., mais bien compliquée - et même peu utilisable - comme modèle d'automate réel.

(z) Ceci a été fait - croyons-nous - aux laboratoires de la Bell Telephone.

un certain nombre d'états. La lecture d'une information sur la bande entraîne, compte tenu de l'état dans lequel se trouve la machine, au moment de la lecture, des actions telles que écriture, mouvement de la bande ou changement d'état.

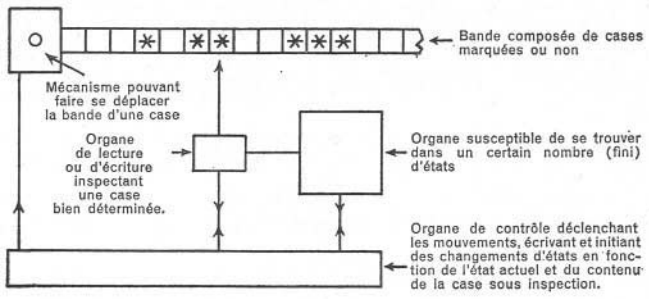


FIG. 11. — Représentation schématique d'une machine de Turing

On peut donc associer à la machine le système formel suivant
L'alphabet comprend les symboles

q_1, q_2, etc., qui correspondent aux « états » ;
S_1, S_2, etc., aux symboles susceptibles d'être lus ou écrits sur la bande;
+, -, aux déplacements possibles de la bande.

Les formules bien formées sont les quadruples de la forme

$$(q_i\ S_j\ S_k\ q_e),\ (q_i\ S_i\ +\ q_e),\ (q_i\ S_i\ -\ q_e),\ (q_i\ S_i\ q_k\ q_e).$$

Les séquences de tels quadruples sont des schémas déductifs sous des conditions d'enchaînement que nous ne donnerons pas ici.
Car les machines à calculer se développèrent dans un contexte technologique tel que le schéma de Turing ne fut pas pris directement en considération. A la place des « quadruples » définissant le système formel, on voit apparaître la notion *d'instruction.* La nuance ainsi introduite apparaîtra clairement en présentant la version « programmationnelle » des machines de Turing telle qu'elle apparaît dans les travaux de H. Wang et C. Lee (i).

(i) Cf. H. WANG, 1. *Assoc. Comp. Mach., 2, ii, 2963.*

Pour ces auteurs, une machine à calculer est un automate qui peut exécuter les instructions suivantes

e : effacer le contenu de la case située dans l'organe de lecture ;
m : marquer la case située dans l'organe de lecture ;
+ : déplacer la bande d'une case vers la gauche ;
- : déplacer la bande d'une case vers la droite ;
t (n) : si la case sous inspection est marquée, sauter jusqu'à l'instruction portant le n° *n,* sinon exécuter l'instruction suivante ;
E (A) exécuter l'instruction A du programme au cas où la case située dans l'organe de lecture est marquée, sinon exécuter l'instruction immédiatement suivante.

Un *programme* est une suite de couples composés d'un numéro d'ordre (séquentiel ou non) et d'une instruction.
Ainsi le programme RTZ *(right to zero)* introduit par Lee est le suivant :

RTZ :
1. +
2. +
3. t (1)

Le lecteur pourra vérifier que ce programme examine une case sur deux, et s'arrête dès qu'il atteint une case non marquée vers la droite.
Bien entendu, un programme tant soit peu complexe comprendra l'exécution répétée de séquences identiques ou semblables. On est donc conduit naturellement à la notion de *sous-programme* ou de macro-instruction. Examinons cela sur un exemple (dû à C. Lee (r)).
Il s'agit d'effectuer, à l'aide de l'automate, le calcul du *carré* d'un nombre entier *n.* Comme la machine de Turing ne comporte aucun organe arithmétique, il nous faut choisir une représentation convenable d'un nombre entier puis reconstituer par programme le processus de l'élevation du carré.
L'état initial et l'état final de la bande, représentant respectivement les entiers n et n^2, seront choisis comme il est indiqué dans la figure *12.*

(i) C. LEE, *Bell Syst. Techn. J.*

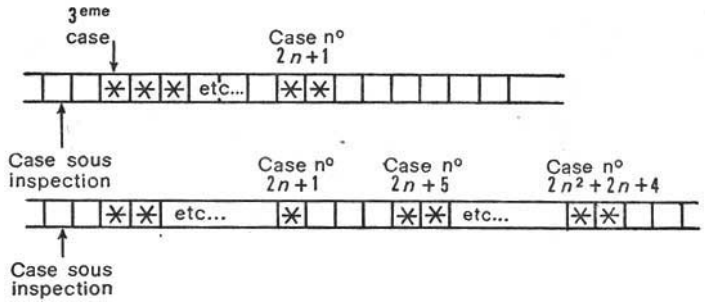


FIG 12 : état initial et état final de la bande
lors d'une élévation au carré dans une machine de Turing

Le programme, sous sa forme originale, est alors assez long à écrire. Par contre, il est aisé de le présenter de la façon suivante

I. + 2, *t (2)*
2. *e,* RTZ, RTZ, *m,* LTZ, LTZ, *m,* + 2, *t (2),* -, *LTZ,* + 2, *t (3)*
3. e, +,LTZ, + 2, *t (2).*

RTZ est défini comme ci-dessus.
LTZ est l'équivalent de

1. –
2. –
3. t (I)

+ 2 est l'équivalent de

1. +
2. +

Si l'on examine un peu plus attentivement le fonctionnement des « macro-instructions », on s'apercevra qu'il faut les écrire, en réalité, sous la forme

i . + (ou -)
i + I. + (ou –)
l + 2. t (i)

où i, i + I et i + 2 sont différents des numéros d'instruction utilisés dans le programme principal.

De même le programme principal ne pourra pas s'écrire simplement

1. $\mathbf{I_1}$
2. I_2
3. $\mathbf{I_3}$

avec

$\mathbf{I_1} = +2, t(2)$
$I_2 = e,$ RTZ, RTZ, $m,$ *LTZ, LTZ,* $m,$ $+2, t(2), -,$ LTZ, $+2, t(3)$
$\mathbf{I_3} = e, +, LTZ, +2, t(2)$

mais bien

I. $\mathbf{I_1}$ (2)
2. I_2 (2, 3)
3. $\mathbf{I_3}$ (2)

avec

$\mathbf{I_1}(n) = +2, t(n)$
$I_2(n_1, n_2) = e,$ RTZ, RTZ, $m,$ LTZ, *LTZ,* $m,$ $+2,$ $t(n_1), -,$ LTZ, $+2, t(n_2)$
$\mathbf{I_3}(n) = e, +,$ LTZ, $+2, t(n).$

Cela veut dire que la mise en place des macro-instructions dans le programme nécessite *un calcul des numéros d'instructions* qui tient compte de la situation particulière de la macro-instruction considérée dans le programme principal.
Cette situation s'éclaire lorsqu'on utilise une représentation graphique de ces expressions formelles : car les macro-instructions RTZ, LTZ, I_1, I_2 et I_3 se représentent par des diagrammes devenant ainsi les éléments d'un diagramme principal, comme il est indiqué dans la figure 13.
Mais au fond comment caractériser le passage de RTZ ou I_2 au programme détaillé qui leur correspond sinon comme une transformation d'un mot écrit dans un certain alphabet en un mot écrit dans un autre alphabet, compte tenu d'un certain nombre de contraintes.
Une telle traduction peut évidemment être faite à la main. Mais il est naturel de construire un programme afin qu'elle s'effectue automatiquement.
D'autre part, de la même façon qu'on a pu trouver une codification des nombres entiers qui en permette la manipulation par l'automate,

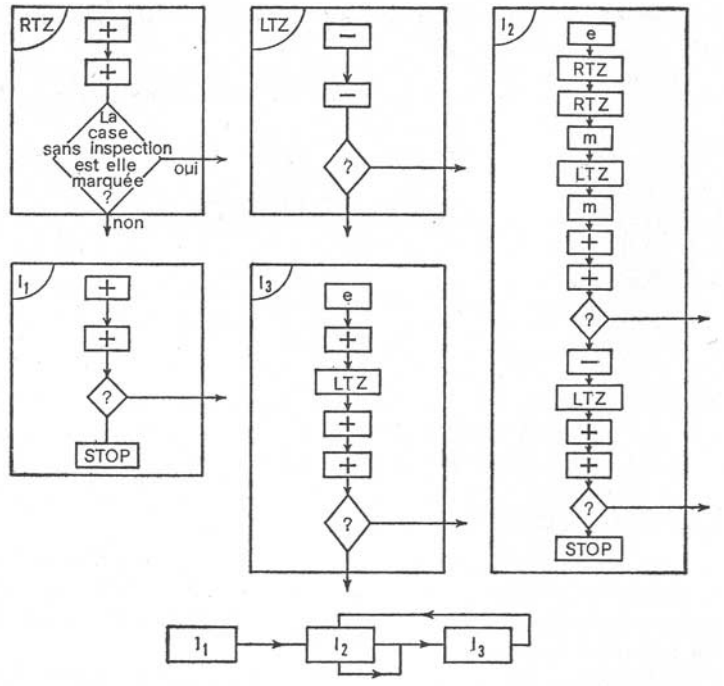


FIG. 13. - Schéma des macro-instructions et du programme principal
Les flèches indiquent l'enchaînement des instructions élémentaires
(ou des macro-instructions)

il est concevable que les instructions elles-mêmes puissent être codifiées et leurs suites manipulées et transformées par l'automate lui-même.

C'est l'idée qui avait été utilisée par Turing dès 1936 pour construire des machines dites « universelles ». C'est ce qui devait être réinventé en 1947 par Burks, Goldstine et von Neumann sous le nom de « programmation enregistrée ».

Ce principe donne immédiatement naissance à une hiérarchie de niveaux que rend la figure 14.

Nous avons abordé la description des automates en passant du langage des systèmes formalisés à celui de la programmation. Cela n'implique aucune modification sensible tant que l'on demeure au niveau syntaxique ou sémantique. Mais au niveau *pragmatique* la

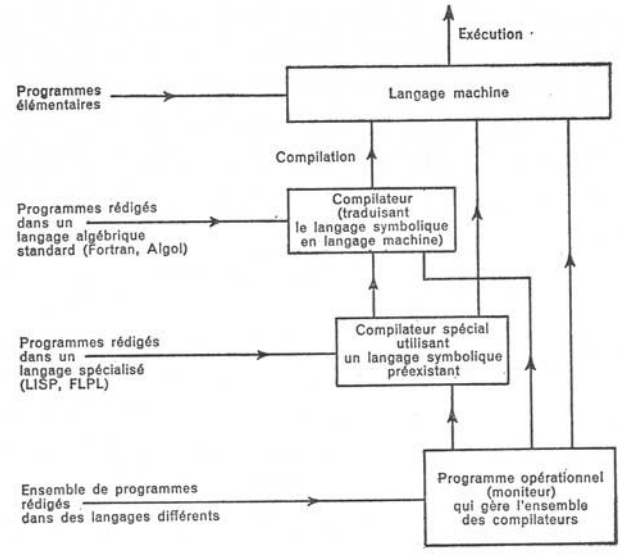


FIG. 14. – Hiérarchie des langages de programmation

différence est considérable. Car l'existence d'un schéma de substitutions du type

$$(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{n-1}, \alpha_n)$$

dans la définition d'un système formel signifie qu'on peut, si on le désire, et si on rencontre une suite d'expressions $\alpha_1, \alpha_2, \ldots, \alpha_{n-1}$, au cours d'une manipulation de symboles la remplacer par l'expression α_n.

Par contre la présence d'une instruction dans un programme impose *l'exécution* de cette instruction au moment où l'automate a fini d'exé-

cuter les instructions qui la précèdent (compte tenu d'éventuels « transferts de contrôle » tels que t (A)).
C'est dire que la manipulation d'un système formel pourra être représentée par un graphe de la forme indiquée dans la figure 15 alors

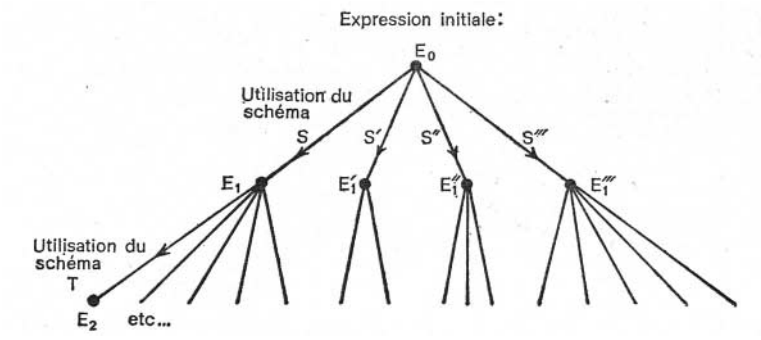
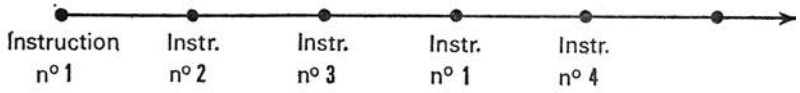


FIG. 55. - Développement « arborescent » d'expressions formelle.
Pour chaque expression, on a le *choix* entre un certain nombre de schémas qui lui sont applicables

que le fonctionnement d'un automate comprendra nécessairement l'accomplissement d'une suite linéaire d'étapes.



Nous aurons l'occasion d'examiner d'autres aspects de cette distinction au cours des chapitres suivants.

***

Bien entendu les automates réels sont infiniment plus compliqués que la machine de Turing.
La *mémoire* n'est pas une simple bande, mais un système à plusieurs niveaux comprenant des organes à accès tridimensionnels (les réseaux de tores à ferrites, par exemple). Les éléments de la mémoire possèdent une *adresse* et il est possible d'y accéder directement, sans passer par un programme fastidieux de déroulement de bande.

Les organes actifs comportent des organes d'addition, arithmétique ou logique, et pas seulement un organe d'écriture et d'effacement, etc.
Nous avons pourtant le sentiment que les notions introduites à l'occasion du modèle de Turing contiennent tout ce qui est nécessaire à la compréhension de ce qui suit. Bien mieux, les machines de Turing ne donnent aucun privilège à l'arithmétique des nombres entiers. Elles sont, par essence, des machines à traiter l'information codifiée sous forme de symboles choisis dans un alphabet. Et c'est tout ce dont nous avons besoin pour aborder les problèmes de l'intelligence artificielle.

***

Nous avons présenté dans ce chapitre sur des exemples particuliers les principes de la formalisation de langages plus ou moins compliqués. Tout d'abord les codes, puis les systèmes logiques et enfin nous avons abordé, avec la théorie des ensembles, des rudiments des systèmes mathématiques.
Le choix que nous avons fait de nos exemples a été guidé par les besoins en outillage formel qui se manifesteront dans les exemples présentés aux chapitres qui suivent.
A cette occasion il est intéressant de faire la remarque suivante
Si les structures mathématiques, et en particulier les structures que l'on définit naturellement au sein du formalisme de la théorie des ensembles, apparaissent comme la mise en oeuvre d'un certain langage formel, les *langages eux-mêmes* sous les diverses représentations que nous leur avons données peuvent être considérés comme des *structures mathématiques particulières.*
Considérons par exemple un alphabet A composé des lettres *a, b, c*, etc. Le fait d'écrire l'une à côté de l'autre les deux lettres *a* et *b* pour former le mot *ab* peut être considéré comme une sorte de multiplication définie sur l'ensemble de toutes les suites de lettres de l'alphabet A ; cet ensemble reçoit d'ailleurs le nom de « monoïde libre » engendré par A. L'opération de multiplication est souvent

appelée « concaténation ». On a donc là une structure algébrique typique.

D'autre part, lorsqu'on utilise une représentation syntaxique des langages comme celle de la figure 6 ou également une représentation de programme sous forme de bloc diagramme comme ceux de la page 44, on utilise un autre type de structure qui est celle des *graphes* qui sont la représentation sagittale de relations binaires. C'est ainsi que la relation binaire définie par l'ensemble des couples *(a, b), (b, c), (a, c), (c, e), (e, a), (a, d), (d, e)* est l'équivalent du graphe de la figure i6.

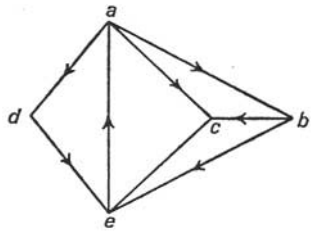


FIG. I6
Représentation « sagittale » d'un graphe

C'est encore là un type bien connu de structure algébrique.

Une seconde remarque est liée à la dualité que l'on peut apercevoir dès à présent entre le langage ou le système formel considéré comme réalisé dans leur totalité et l'automate muni d'un programme qui engendre les expressions, les formules de ce langage, au fur et à mesure de son fonctionnement. Nous aurons l'occasion de revenir à plusieurs reprises sur cette distinction qui est celle de *l'extension* et de *l'intension.*

CHAPITRE III

Les jeux

Dans le chapitre VIII au cours duquel nous rassemblerons un certain nombre d'informations d'ordre historique, on verra que, même chez les lointains précurseurs de l'intelligence artificielle, la simulation des jeux a été considérée comme un exemple ou un test significatif.

Cela n'est pas étonnant si l'on adopte, pour aborder le problème de l'intelligence, un point de vue génétique analogue à celui de Jean Piaget, comme nous avons tenté de le faire au cours du premier chapitre.

L'activité ludique commence avant même l'activité linguistique et se mélange intimement avec elle. Elle met en oeuvre les six stades que nous avons énumérés en analysant les étapes de l'apparition de l'intelligence, en particulier la troisième (procédés destinés à faire durer les spectacles intéressants) et la cinquième (découverte des moyens nouveaux par expérimentation active).

C'est donc dans le jeu que s'exerce l'intelligence naissante. Non contente de l'apprentissage que la rencontre du monde lui impose, elle se donne ainsi la possibilité d'une expérimentation accélérée où les problèmes de l'univers réel sont posés sous une forme concentrée.

La simulation des jeux est ainsi un exercice privilégié pour la recherche des méthodes et pour l'estimation des difficultés en intelligence artificielle.

D'ailleurs si l'automate ne peut manipuler que des informations préalablement codées, et ceci en fonction d'un programme lui-même rédigé de façon convenue, il y a évidemment intérêt à se proposer tout d'abord des « textes » tirés d'un langage assez pauvre, que ce soit du point de vue du vocabulaire ou de celui de la syntaxe.

Ici encore, les jeux nous fournissent une entrée en matières commode. Tout comme dans le cas d'un système formel de type mathématique,