LA SCIENCE VIVANTE

Collection dirigée par **HENRI LAUGIER**

Pierre AUGER. — **RECHERCHE ET CHERCHEURS SCIENTIFIQUES** ..... 7 F.
Henri PRAT. — **LE CHAMP UNITAIRE EN BIOLOGIE** ..... 12 F.
Michel-Yves BERNARD. — **MASERS ET LASERS** *(2e édition)* . 15 F.
Vladimir KOURGANOFF. — **INITIATION A LA THÉORIE DE LA RELATIVITÉ** ..... 12 F.
D. M. MAIN. — **INTRODUCTION A LA PHYSIQUE NUCLÉAIRE** ..... 12 F.
Maurice AUBERT. — **CULTIVER L'OCÉAN** ..... 16 F.
Robert GIBRAT. — **L'ÉNERGIE DES MARÉES** ..... 18 F.
Jean DALSACE et Raoul PALMER. — **LA CONTRACEPTION** *(3e édition revue)* ..... 12 F.
Cyril GOMELLA. — **LA SOIF DU MONDE ET LE DESSALEMENT DES EAUX** ..... 16 F.
Marthe BONVALLET. — **SYSTÈME NERVEUX ET VIGILANCE** . 10 F.
Michel SABOURDY. — **L'ANIMAL DE LABORATOIRE DANS LA RECHERCHE BIOLOGIQUE ET MÉDICALE** ..... 18 F.
Yves LE GRAND. — **LUMIÈRE ET VIE ANIMALE** ..... 12 F.
Henry G. ROBERT. — **LES ORGANES ARTIFICIELS** ..... 12 F.
Francis DELOBEAU. — **L'ENVIRONNEMENT DE LA TERRE** . 15 F.
Michel GRENON. — **LE TRAVAIL EN MILIEU HOSTILE** .... 18 F.
André DJOURNO et Danièle KAYSER. — **ANESTHÉSIE ET SOMMEIL ÉLECTRIQUES** ..... 12 F.
Evry SCHATZMAN. — **PLASMAS ET MILIEUX IONISÉS** ..... 16 F.
Paul BRAFFORT. — **L'INTELLIGENCE ARTIFICIELLE** ..... 15 F.
Jean A. TERNISIEN. — **LES POLLUTIONS ET LEURS EFFETS** ..... *(sous presse)*
Jean A. TERNISIEN. — **LA LUTTE CONTRE LES POLLUTIONS** ..... *(sous presse)*
Henri DESSENS. — **LA MAITRISE DES CLIMATS** ... *(sous presse)*
Jean-Claude PECKER. — **ASTRONOMIE ET RECHERCHE SPATIALE** ..... *(sous presse)*

PRESSES UNIVERSITAIRES DE FRANCE

29 878. C

PAUL BRAFFORT

L'INTELLIGENCE ARTIFICIELLE

La Science Vivante

PRESSES UNIVERSITAIRES DE FRANCE

L'INTELLIGENCE ARTIFICIELLE

DU MÊME AUTEUR

Computer Programming and formal systems (édité en collaboration avec D. HIRSCHBERG), North-Holland, 1962.

« *La Science Vivante* »

Collection dirigée par Henri LAUGIER

Professeur honoraire à la Sorbonne

L'INTELLIGENCE ARTIFICIELLE

par

PAUL BRAFFORT

Préface de Jules GUERON

Directeur général des recherches à la Communauté Européenne de l'Énergie Atomique
Ancien Directeur au Commissariat à l'Énergie Atomique

PRESSES UNIVERSITAIRES DE FRANCE
108, BOULEVARD SAINT-GERMAIN, PARIS

1968

DÉPÔT LÉGAL

I^re édition . . . . . 2e trimestre 1968

Ce travail est dédié à la mémoire
de

E. W. BETH
Professeur à l'Université d'Amsterdam
(1908-1963)

Préface

En présentant à Henri Laugier Paul Braffort, j'espérais bien que la collection « La Science vivante » s'enrichirait d'un livre.

Mais l'amitié que l'un et l'autre veulent bien me témoigner justifie seule qu'ils m'aient demandé d'écrire une Préface - et que je me risque à le faire.

Car je ne puis nullement juger - donc présenter - en connaisseur ce texte que j'ai lu avec un intérêt et un plaisir très vifs.

A cause, en premier lieu, du sujet même, et de l'impression que le livre me donnait de le comprendre.

Mais aussi à cause du souvenir d'une tentative qui remonte à près de dix ans. Car, en préparant l'implantation du Centre Commun de Recherches d'Euratom (qui devait plus tard s'installer à Ispra), nous étions convaincus que si le gaz neutronique dont sont emplis nos réacteurs exigeait, pour être bien compris, une grande calculatrice, il ne devrait pas l'occuper seul.

Comme certains autres, mais avant beaucoup, nous voulions nous attaquer au traitement de l'information en général, et pas seulement de façon empirique.

Comme tous les scientifiques la marée de l'information nous submergeait. Comme membres de la Communauté Européenne, nous ressentions de façon aiguë la multiplicité des langues.

Et, témoins de l'évolution rapide des machines, nous ne concevions pas que les utilisateurs puissent ne pas intervenir dans la transformation de l'instrument.

D'où la constitution rapide d'une équipe bouillonnante, productive (comme le montreront au lecteur attentif diverses indications discrètes des pages qui suivent) et peut-être un peu bohème - intellectuellement parlant.

Mais certains estimèrent que, dans un centre nucléaire, on ne devait rien faire (sinon accessoirement) que développer des réacteurs. D'autres, au nom de la liberté de la recherche, et de celle de l'industrie, soutenaient que les réacteurs eux-mêmes n'avaient point de place dans de tels centres. Mais tous nous firent bientôt voir que, surveillée ou prônée, la liberté doit être timide et conventionnelle - intellectuellement parlant.

Coïncidant avec ce double conformisme et avec les tourbillons qui commençaient à tenter de dissocier l'Europe à peine assemblée, « l'explosion de complexité », si bien décrite au chapitre VI, « souffla » l'équipe d'Ispra, projetant de l'atome à l'espace son animateur, l'auteur de L'intelligence artificielle.

Le chapitre VII nous assure que l'intelligence, artificielle ou non, ne s'arrête jamais, et que, si subtils soient-ils, les « impossibilistes » sont aujourd'hui moins convaincants - en tout cas moins suivis - qu'il y a 5 ans. J'espère que les éditions successives de ce livre en apporteront la preuve toujours plus claire.

Jules GUERON,
Directeur général des Recherches
à la Communauté Européenne de l'Energie Atomique,
Ancien Directeur au Commissariat à l'Energie Atomique.

Avant-Propos

Afin de ne pas trop alourdir le texte, nous avons inclus dans le chapitre VIII, en fin de volume, un *Who's who* des principaux chercheurs dans le domaine de l'intelligence artificielle, ainsi qu'une bibliographie commentée.

Par contre, les références à des travaux non cités dans cette bibliographie sont données en bas de page. On y trouve aussi le rappel de quelques définitions utiles à la compréhension du texte.

Il est possible que, malgré ces rappels, des lecteurs éprouvent quelques difficultés en certains paragraphes un peu techniques. Il nous a semblé préférable d'affronter ce danger plutôt que de nous maintenir dans des généralités inoffensives, en raison même du sujet que nous avons choisi : car les discours y ont pendant trop longtemps remplacé les actes, c'est-à-dire les travaux scientifiques concrets.

Introduction

« Intelligence artificielle » : c'est là une expression qui, pour le non-spécialiste, évoque une constellation d'images, d'idées, souvent proche du fantastique. Les mots clés : « robot », « cerveau électronique », « machine à penser », surgissent, dans une ambiance de science-fiction, un cliquetis d'engrenages, une odeur d'ozone : Frankenstein n'est pas loin !

A un niveau de conscience plus claire, plus rationnelle, on évoque les dangers de l'automation : chômage, surproduction et autres sources de confusion économique et sociale.

Enfin, plus près des centres de responsabilités, au niveau des universitaires, des industriels, des gouvernants, on se préoccupe de contrôler des techniques, des méthodes, et surtout des potentialités dont on soupçonne l'importance.

Chaque grand pays s'efforce de conserver ou d'acquérir une indépendance jugée précieuse dans l'industrie des machines à calculer électroniques, de développer la formation de mathématiciens, logiciens et autres spécialistes indispensables à l'utilisation efficace des machines, d'attirer les chercheurs dans un domaine nouveau dont les frontières sont loin d'être encore visibles.

* * *

Notre propos est de permettre au public cultivé de passer du niveau des rêveries frankensteiniennes à celui d'une actualité scientifique et technique récente... Ceci n'implique, bien entendu, aucune condamnation de la littérature de « science-fiction ».

Mais pour mieux permettre à l'imagination, à la rêverie poétique de

colorer la progression tâtillonne et maussade des techniques, il est indispensable de mieux connaître celles-ci, et, si possible, de les précéder un peu.

***

Quittant donc la légèreté du romanesque pour le cheminement souvent laborieux des constatations et des déductions, nous ne saurions imaginer meilleur « baptême du froid » que celui que nous propose la bibliographie scientifique. « Intelligence artificielle», c'est en effet, depuis longtemps déjà, le «mot clé » caractéristique de rubriques qui, dans les revues de résumés analytiques spécialisées, rassemblent, à chaque publication, des dizaines d'articles publiés dans le monde entier.

En principe, il suffirait donc d'examiner le contenu de telles rubriques au cours des dernières années pour avoir une bonne idée de ce qu'est l' « intelligence artificielle ».

Pour le vérifier, nous allons prendre un exemple récent que nous avons choisi complètement au hasard.

***

Dans le numéro d'avril 1966 de la revue *Transactions of the Institute of Electrical and Electronic Engineers on Electronic Computers, plus* brièvement *IEEE-EC 15,* on trouve, comme dans chaque numéro, une revue bibliographique. En particulier, il existe une rubrique « Behavioral sciences and artificial intelligence ». A la page 285 de ce numéro, se trouve le résumé no 4349 ; il s'agit d'un article publié dans la revue *Communications of the Association for Computing Machinery* en décembre 1965 (tome 8, p. 792), par J. R. Slagle.

Nous présentons ci-dessous une traduction de l'analyse, de cet article, telle qu'elle est offerte aux lecteurs des *Transactions*

Expériences avec un programme déductif permettant de répondre à des questions

Dans le cadre de recherches sur l'intelligence artificielle, on a conduit quelques expériences relatives à un programme pour calculatrice, appelé DEDUCOM (pour DEDUctive COMmunication), programme déductif permettant de répondre à des séries de questions. Ayant enregistré 68 «faits », DEDUCOM

a pu répondre à 1o questions dont les réponses mettaient en ceuvre les faits qui avaient été fournis.

Un « fait », donné à DEDUCOM est, soit une information spécifique, soit une méthode pour répondre à une classe de questions. Parmi les conclusions que l'on trouve dans l'article, citons

1) DEDUCOM peut répondre à une grande variété de questions.
2) L'homme peut accroître le pouvoir déductif en lui donnant davantage de « faits ».
3) DEDUCOM peut rédiger des programmes très simples (on espère que cette capacité présage l'autoprogrammation qui est une voie vers l'apprentissage).
4) DEDUCOM répond très lentement aux questions posées.
5) La procédure de recherche de DEDUCOM a pour le moment deux défauts principaux

- certaines questions auxquelles il devrait être normalement possible de répondre n'obtiennent pas de réponse ;
- certaines autres n'obtiennent de réponse que si les faits auxquels elles se rapportent ont été présentés dans un ordre « convenable ».

6) Pour le moment, la méthode utilisée par DEDUCOM pour effectuer des déductions logiques suivant le calcul des prédicats présente aussi deux lacunes sérieuses

- certains faits doivent être transformés en faits équivalents avant d'être introduits dans le programme ;
- certains faits redondants doivent être fournis.

L'intérêt de cette analyse - tout au moins en ce qui nous concerne ici - est d'introduire d'un seul coup un grand nombre des notions que nous allons étudier en détail par la suite : on peut en effet en inférer que l'intelligence artificielle s'intéresse aux « machines à calculer » et à leur « programmation », qu'il y est fait usage de « processus déductifs », et en particulier du « calcul des prédicats » et que les objets manipulés par les machines ainsi programmées peuvent être d'une essence non directement mathématique : « faits », « questions » et « réponses ».

***

Nous aurions sans doute pu prendre une vingtaine d'articles analysés dans les périodiques spécialisés et accumuler ainsi un certain nombre de notions, de « mots clés » que nous aurions pu ensuite regrouper, puis expliquer progressivement.

Il nous a semblé plus prudent, dans un domaine où les mauvaises interprétations sont encore nombreuses, de procéder de façon plus dogmatique, quitte à retarder de quelques chapitres l'examen de l'intelligence artificielle « militante ».

C'est ainsi que nous mettrons face à face, dans un premier chapitre, le domaine de *l'intelligence* et celui des *automates.*

Car le domaine de l'intelligence « naturelle », c'est en effet celui de la psychologie, de la psychophysiologie, de la psychopathologie, etc. C'est un domaine immense qui couvre de vastes régions telles que la «mémoire», l' «imagination», la «raison», etc., qui, bien que mal définies et délimitées, ont fait l'objet de recherches sans nombre, tant au point de vue théorique qu'au point de vue expérimental.

Le domaine de l'intelligence «artificielle», tel qu'il a été évoqué dans les pages qui précèdent, est infiniment plus restreint. Il s'agit de l'utilisation de machines à calculer électroniques pour une classe de problèmes importante, mais cependant spécifique.

On peut donc se demander si l'utilisation du même mot « intelligence» dans les deux cas n'est pas le résultat d'un abus de langage, et la source possible de dangereuses confusions.

Dans le chapitre I nous pensons montrer qu'il n'en est rien et qu'un domaine fondamental est commun à l'intelligence et aux automates celui de la manipulation de codes organisés, de langages, et ceci selon une hiérarchie complexe de niveaux d'organisation.

Ceci nous amènera à consacrer un chapitre entier, le chapitre II, à une étude rapide des codes, systèmes formels, langages de programmation. Nous disposerons alors de l'outillage nécessaire pour aborder les rubriques de l'intelligence artificielle proprement dite.

Notre progression sera plus didactique qu'historique ou logique car, si toutes ces rubriques sont liées, elles se sont développées simultanément et ont observé un certain parallélisme, tant méthodologique que technique.

Nous débuterons, dans le chapitre III, par l'étude des jeux, ou, plus précisément, de la construction de programmes pour machines à calculer qui imitent le comportement d'un joueur. Les jeux s'expriment en effet à l'aide de « langages » dont le vocabulaire et les règles sont relativement « pauvres ». Dès ce niveau, on verra pourtant les formidables difficultés que doivent affronter les chercheurs.

Puis, dans le chapitre IV, nous aborderons l'imitation par la machine du raisonnement mathématique. Ici le vocabulaire demeure restreint, mais les règles se compliquent.

Enfin, dans le chapitre V, nous atteindrons le niveau du langage humain et de son traitement automatique par les machines à calculer.

A cette progression correspond, on s'en doute, une progression de la difficulté pour les chercheurs.

D'ailleurs un mot reviendra souvent dans les commentaires : celui de *complexité.* Mais cette notion dont on pourrait se servir comme d'une excuse devant certains échecs devient au contraire, si on l'aborde de front, l'objet d'une définition et d'une étude scientifiques, la clé d'une unification des méthodes et d'un progrès décisif. Aussi sera-t-elle le sujet de notre chapitre VI.

Nous terminerons en élargissant notre champ de vision pour situer les problèmes de l'intelligence artificielle dans le contexte actuel du traitement automatique de l'information afin de donner des perspectives d'avenir convenablement fondées. Ce sera l'objet du chapitre VII.

Enfin le chapitre VIII nous donnera l'occasion d'incarner nos propos en insistant davantage sur l'aspect historique et même individuel des recherches et des découvertes.

***

Tout au long du texte, nous multiplierons les exemples qui, en général, ne correspondent pas à des problèmes réels tirés de recherches effectivement poursuivies, mais ont été construits spécialement pour illustrer notre exposé sans exiger un apport supplémentaire de concepts et de méthodes.

Le plan que nous venons de présenter ne permettra donc qu'au bout de plusieurs chapitres de bien comprendre le contenu de l'article que nous citions au début de cette introduction sur le programme « DEDUCOM ».

Mais il nous semblait important de souligner, aux premières lignes de cet ouvrage, que l'on traitera ici d'une discipline bien vivante, et non pas de recherches futures, problématiques.

CHAPITRE PREMIER

Intelligence et artifices

Lorsqu'on met en regard les pouvoirs de l'intelligence et ceux de constructions humaines comme les automates, qui n'en sont qu'un reflet bien partiel, on s'attire inévitablement des objections passionnées de même qu'on suscite quelques enthousiasmes naïfs. Le mot même d'automatisme est souvent utilisé comme le *contraire* de l'acte intelligent. Ceci est vrai jusqu'aux niveaux les plus élevés de la vie mentale de l'individu où il est commode d'opposer par exemple l'activité créatrice du mathématicien au comportement réflexe du calculateur prodige.

Il est donc indispensable, au début de cet ouvrage, de rappeler, en quelques paragraphes, ce que sont les conditions réelles de la naissance et du développement de l'intelligence afin d'en esquisser les structures essentielles - dans la mesure encore limitée où il est possible de les apprécier aujourd'hui.

Ceci fait, on examinera succinctement, mais également d'un point de vue « du comportement », la naissance et le développement des « artifices » divers qui peu à peu enrichissent notre arsenal automatique. On verra alors apparaître un domaine commun dont l'examen plus approfondi fera l'objet du chapitre II.

LES OPERATIONS DE L'INTELLIGENCE

L'intelligence, telle qu'elle se manifeste chez les animaux supérieurs et notamment chez l'homme, ne se laisse pas commodément cerner. On admettra cependant qu'une de ses caractéristiques essentielles est

cet aspect de circularité qu'elle présente, au bout d'une longue évolution ; circularité qui se manifeste dans le fait qu'elle seule, parmi les acquisitions successives des espèces, jouit de la propriété d'être « réflexive », c'est-à-dire de pouvoir s'appliquer à soi-même.

On peut en effet « penser à sa propre pensée » alors qu'on ne peut pas « voir l'acte de voir », qu'on ne peut pas «lancer un jet », etc.

Mais avant que de posséder cette vertu réflexive, l'intelligence est une acquisition à un double titre ; celui de l'espèce et celui de l'individu. Il est donc naturel, pour en éclairer quelques aspects essentiels, de faire appel au point de vue « génétique » qui s'efforce d'en saisir et d'en articuler les premiers balbutiements.

C'est ainsi qu'on trouve, sous la plume de Jean Piaget (I), la définition suivante

> L'intelligence est une adaptation. Pour saisir ses rapports avec la vie en général il s'agit donc de préciser quelles relations existent entre l'organisme et le milieu ambiant. En effet, la vie est une création continue de formes de plus en plus complexes et une mise en équilibre progressive entre les formes et le milieu. Dire que l'intelligence est un cas particulier de l'adaptation biologique, c'est donc supposer que telle est essentiellement une organisation et que sa fonction est de structurer l'univers comme l'organisme structure le milieu immédiat.

Le mot *organisation* est celui sur lequel nous voudrions attirer particulièrement l'attention. Car il implique deux acceptions également possibles, et que d'ailleurs nous utiliserons toutes deux : organisation en tant que *système* d'opérations, organisation en tant que *résultat* de ce système d'opérations.

Car si l'intelligence caractérise certaines de nos activités supérieures, celles-ci peuvent présenter aussi bien un aspect *passif,* correspondant à une assimilation de messages venus de l'extérieur, qu'un aspect *actif,* en tant qu'interventions de l'individu dans le milieu qui est le sien. Le rôle de l'intelligence, c'est, en gros, d'assurer le succès de ces interventions, succès qui se situe à des niveaux de satisfaction très variés, mais qui, en fin de compte, mesurent l'adaptation de l'individu à son milieu.

(I) J. PIAGET, La naissance de l'intelligence chez l'enfant, Delachaux & Niestlé, 1959, p. I0.

Lorsqu'elle atteint son développement normal chez l'adulte contemporain, l'intelligence peut en effet être caractérisée comme une aptitude à manipuler les *concepts* en vue d'actions efficaces dans le cadre d'une finalité prédéterminée.

Mais ce niveau n'est évidemment que l'aboutissement d'un long parcours dans l'évolution de l'espèce humaine, et, au moins par certains côtés, de l'évolution tout court.

Car l'homme (comme déjà l'animal), aux prises avec le monde extérieur, ne peut pas *s'approprier* les objets dont ce monde est fait. Il ne peut les connaître et les reconnaître qu'au travers d'une *représentation* dont la forme la plus simple est la sensation. Les sensations, trop nombreuses pour être exploitables directement (cet obstacle des grands nombres sera l'objet principal des discussions de notre chapitre VI), sont elles-mêmes classées en groupes et en familles de groupes, etc.

Ces regroupements s'opèrent probablement à l'aide de critères d'invariance par rapport à des opérations élémentaires (déplacement de l'observateur, action d'agents physiques, etc.), ou à des schémas opératoires que nous ne nous proposons évidemment pas de préciser davantage ici.

Même sous cette forme très simplifiée, il saute aux yeux que l'intelligence est un phénomène excessivement complexe. Car tant que le monde extérieur ne nous apparaît que comme une suite aléatoire de sensations, il n'y a aucun espoir de construire un comportement adapté. L'intelligence doit donc disposer

- d'un ensemble de classifications qui permettent d'établir des ressemblances et de vérifier des cohérences ;
- d'une possibilité de stocker la masse d'informations ainsi codifiée de façon élémentaire.

Cette double nécessité se situe elle-même :

- au niveau des rapports entre l'individu et le monde extérieur ;
- au niveau des rapports que les individus entretiennent entre eux.

Tout ceci engendre une hiérarchie de technique d'abstraction et de mémorisation, hiérarchie qui comprend notamment le langage - écrit et parlé -, l'activité rationnelle - y compris l'activité scientifique

et technique - et ceci jusqu'à des niveaux aussi raffinés que ceux de la mathématique et de la logique formelle.

Pour tenter d'apporter quelque lumière dans cet enchevêtrement constructif, replaçons-nous dans l'optique « génétique » qui est celle de Jean Piaget. Cet auteur distingue six *stades* dans l'apparition de l'intelligence

- l'exercice des réflexes ;
- les premières adaptations acquises (réaction circulaire primaire) ;
- les réactions circulaires secondaires et les procédés destinés à faire durer les spectacles intéressants ;
- la coordination des schémas secondaires ;
- la réaction circulaire tertiaire et la découverte des moyens nouveaux par expérimentation active ;
- l'invention des moyens nouveaux par combinaison mentale.

Comme l'indique d'ailleurs Jean Piaget (I) :

Du simple réflexe à l'intelligence la plus systématique, un même fonctionnement nous paraît se prolonger au travers de tous les stades, établissant ainsi une continuité entière entre des structures de plus en plus complexes. Mais cette continuité fonctionnelle n'exclut en rien une transformation des structures allant elle-même de pair avec un véritable renversement des perspectives dans la conscience du sujet. Au début de l'évolution intellectuelle, en effet, l'acte est déclenché tout d'une pièce et par un stimulus extérieur, l'initiative de l'individu consistant simplement à pouvoir reproduire son action en présence d'excitants analogues au stimulus normal, ou par simple répétition à vide. Au terme de l'évolution, au contraire, toute action implique une organisation mobile à dissociations et regroupements indéfinis, le sujet pouvant ainsi s'assigner à lui-même des buts toujours plus indépendants de la suggestion du milieu immédiat.

Ce qui nous semble essentiel dans tout ceci, en vue des comparaisons que nous nous proposons d'effectuer dans la suite, c'est l'existence d'un système de communication entre l'être intelligent et le monde extérieur, communications rendues possibles par un jeu complet de *substitutions*

— substitutions de sensations aux objets extérieurs ;
- substitution d'excitations cérébrales aux messages sensoriels ;

(I) Loc. cit., p. 137.

— substitution de « souvenirs » à des classes d'expériences ;
- substitution de systèmes à des séquences de souvenirs isolés ;
- substitution de concepts à des classes de systèmes ;
- etc.

Il est également remarquable que les reconstitutions du processus d'acquisition de l'intelligence du type de celui que nous venons d'évoquer s'appuient sur des descriptions « de l'extérieur » et ne font intervenir en rien d'essentiel le *support physiologique* de l'activité intellectuelle.

Présentée ainsi en termes de comportement, l'intelligence semble si bien indépendante du support matériel qui lui permet de s'exercer qu'on en vient naturellement à imaginer qu'un changement de support serait aisément réalisable, allant par exemple du physiologique au mécanique, afin d'effectuer une *simulation plus* ou moins complète que nous allons justement chercher à décrire. Mais puisque jusqu'à présent cette recherche n'a pas rencontré que des succès, il n'est pas mauvais de dresser, face au tableau des exigences de l'intellectuel, un bilan de l'évolution des artifices, des machines, des robots que l'homme, peu à peu, invente, pour leur confier un nombre toujours plus grand de tâches.

MACHINES ET AUTOMATES

Tout comme l'intelligence qui, dans l'évolution des espèces ou dans la vie de l'individu, n'apparaît pas soudain *ex nihilo,* mais prend forme au long d'une série d'étapes, les machines sont, elles aussi, le produit d'une évolution qui se fait bien entendu selon une autre échelle de temps et dans un contexte bien spécifique.

Les *outils* les plus primitifs sont, on le sait, des substitutions ou des amplifications d'organes humains - en particulier des mains, en vue d'une action plus efficace sur le monde extérieur. L'amplification en question est de nature qualitative (par exemple dans le sens du tranchant ou du pointu) mais la force et le travail sont fournis entièrement par l'individu (I).

(I) Voir par exemple A. Laaot-Goulus x, *L'homme et la matière,* Albin Michel.

Les *machines* apparaissent dès qu'il est possible de stocker le travail humain, ou même de lui substituer d'autres efforts (animaux ou sources naturelles inanimées).

Dans toute cette phase du développement, la part « intelligente » de ces transformations de la nature reste sous le contrôle actif de l'individu qui décide de l'enchaînement des actions physiques qu'il confie aux mécanismes.

Lorsque les décisions peuvent être codifiées, stockées et enchaînées tout comme les actions mécaniques, on entre dans le règne de l'automatisme.

Tout à la fin apparaît - et cette histoire devient la nôtre - l'automatisme appliqué non plus à des actions mécaniques comme le contrôle des machines-outils, des trains et des voies, des circuits industriels, etc., mais aux informations elles-mêmes, sous leur forme la plus abstraite, même mathématique.

Car les automates ont été longtemps des mécanismes au comportement entièrement déterminé par une tâche particulière et permettant ainsi d'économiser une ou plusieurs actions humaines. Toute l' « intelligence » impliquée dans la tâche automatisée était incarnée dans la structure même du mécanisme.

C'est ainsi que le distributeur « automatique » ne répond qu'à une séquence bien déterminée d'instructions

a) introduire dans la fente une pièce de monnaie (de valeur fixée) ;
b) tirer sur une poignée,

et en réponse à cette excitation standard, il fournit une réponse unique.

Mais, peu à peu, on s'est *efforcé* de construire des automates qui acceptent des conditions variables et y répondent en fonction d'un certain nombre de consignes. Il fallait donc exprimer ces consignes et les informations relatives aux conditions variables dans un même langage. Par exemple une machine à laver automatique possède des consignes telles que : « rincer », « essorer », etc., des conditions variables : « température de l'eau», « temps écoulé », etc., qui, pour l'automate, deviennent des positions de relais et de cames, permettant de *programmer* le fonctionnement sous forme d'une articulation convenablement choisie des consignes et des conditions.

Cette évolution s'est poursuivie - au moins pour les automates complexes (distilleries, laminoirs, etc.) - jusqu'au point où les actions physiques ont été confiées à des mécanismes passifs ordinaires, mais commandés par des signaux provenant d'un automate central, machine à calculer électronique à peine spécialisée, qui traite les informations provenant d'organes de mesure et prend les décisions actions après divers calculs ou contrôles logiques. Cette organisation n'est possible que grâce à l'existence de « transducteurs » qui traduisent les données les plus variées (pressions, températures, nombres d'impulsions, etc.) dans un langage standard qui est celui de la calculatrice. Ici encore on a un système de substitution dont on verra plus loin qu'il donne naissance lui aussi à une hiérarchie qui peut atteindre une certaine complexité.

Pour se faire une idée du chemin parcouru, on peut citer l'exemple de l'ensemble d'automates, portant le nom *d'Instrument Unit,* qui a été commandé par la N.A.S.A., l'organisation spatiale américaine, à I.B.M., le plus important constructeur mondial de machines à calculer. Cet ensemble est destiné à constituer le centre vital de la fusée *Saturne,* qui devra propulser vers la lune les satellites du projet *Apollo.* Soixante sous-systèmes électroniques le constituent afin de guider et de contrôler à tous moments la fusée.

L' I.U. traite des millions d'informations par minute.

Avant le lancement, l'I. U. effectue les contrôles de routine de la fusée et se contrôle lui-même.

Pendant le lancement, *l' I. U.* assure le déclenchement et l'enregistrement d'innombrables mesures (notamment vitesse, accélération, etc.) et en assure la transmission aux stations à terre. Après la mise en orbite, cette même unité commande le départ vers la lune puis la séparation du « module » d'exploration lunaire.

Cet énorme automate est mis au point dans un centre spécial où travaillent i 8oo personnes. On voit sans peine le chemin parcouru depuis le distributeur automatique de paquets de cigarettes (I) !...

(I) On trouvera des indications générales intéressantes sur l'automatisme dans le livre de P. de Latil, et sur un plan plus technique dans le livre de F: H. Raymond, tous deux cités dans la bibliographie.

***

Cette brève histoire des machines offre à la fois similitudes et contrastes avec celle de l'intelligence. Mais le premier aspect que nous désirons souligner ici, c'est l'absence de tout anthropomorphisme dès l'invention des mécanismes et des automations. On utilise des matériaux dont les qualités de robustesse et de longévité sont bien supérieures à celles des matériaux dont nous sommes faits. On invente des systèmes - comme la roue - dont nous sommes dépourvus. Finalement on fabrique des organes - comme la mémoire - dont l'équivalent humain nous est encore inexpliqué.

Cette constatation devrait à la fois expliquer les difficultés de l'intelligence artificielle et son caractère strictement technique.

Et c'est dans le souci d'éviter toute polémique renouvelée de la scholastique (comme celle qui fleurit en ce moment dans les pays anglo-saxons sous le nom de *mind-body problem (I))* que nous avons aussi déterminé le cours de notre exposé.

Beaucoup de bons esprits se préoccupent en effet de savoir jusqu'à quel point une machine peut « réellement penser», comment s'effectue l' « émergence » d'un psychisme à partir du support matériel qui le conditionne, etc.

Par ailleurs, d'autres chercheurs s'efforcent, en construisant des modèles de neurones ou d'assemblées de neurones, d'acquérir des lumières sur ce que pouvait être le fonctionnement du cerveau, son activité d'apprentissage et de création.

En ce qui nous concerne, nous nous bornerons à constater le divorce qui - au moins pour le moment - existe entre le support matériel de l'intelligence humaine et celui de l'intelligence artificielle, et nous nous abstiendrons de tirer des résultats que nous citerons toute conclusion d'ordre psychophysiologique ou métaphysique. D'ailleurs nous bannirons délibérément des expressions équivoques, telles que « cerveaux électroniques », « machines à penser », etc.

C'est aussi pour ces diverses raisons que nous avons été conduits à éliminer de notre enquête les études d'automation plus ou moins

(I) Cf. l'ouvrage collectif édité par S. Hook (cité en bibliographie).

anthropomorphistes, robots, tortues et renards qui - au détriment d'ailleurs des performances intelligentes - singent le mécanisme extérieur des êtres animés.

LES MACHINES A CALCULER

Plus précisément nous nous limiterons aux recherches qui se fondent sur *l'utilisation de machines à calculer universelles* pour la simulation des tâches intellectuelles.

Il est clair que l'expression « machine à calculer », même si elle est plus prosaïque que celle de « cerveau électronique », éveille encore dans le public un sentiment d'admiration craintive. Il est temps de s'en défaire et pour cela il faut se rendre compte qu'aujourd'hui les machines à calculer sont présentes partout (ceci est vrai dans notre pays, mais naturellement encore plus vrai aux États-Unis).

Pour fixer les idées citons quelques chiffres empruntés à un exposé de S. Fernbach (*I*) au récent congrès de l'I.F.I.P. (Fédération Internationale pour le Traitement de l'Information).

Selon cet auteur, la distribution actuelle des machines à calculer aux États-Unis est donnée figure i.

Dans une estimation plus récente publiée par *Scientific American (2),* J. McCarthy précise que si le nombre de calculateurs utilisés aux États-Unis en 1950 était de l'ordre de 15 il est aujourd'hui voisin de 32 000 et doit passer à 85 000 en 1975. On prévoit qu'à cette époque les investissements en machines à calculer atteindront 30 milliards de dollars, c'est-à-dire *150 milliards de nos francs actuels !*

***

Les machines à calculer ne peuvent donc plus raisonnablement être considérées comme des bêtes curieuses. En particulier leur architecture ne devrait présenter, même pour le profane, aucun mystère. Nous n'avons pas l'intention de reprendre dans ce livre une description

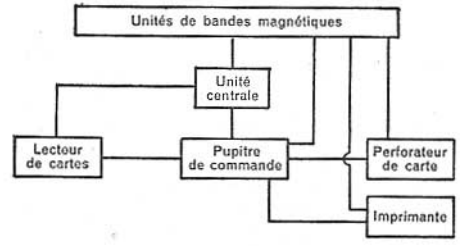
(I) Cf. S. FERNBACH, Computers in the U.S.A. : to-day and to-morrow, Proc. of *I.F.I.P. Congres, 65,* vol. I, p. 77.

(2) Cf. numéro spécial cité en bibliographie.

Classe	Principaux constructeurs	Nombre de types dans cette classe	Machines installées	Machines en commande	Investissement actuel (en millions de dollars)	Investissement prévu (en millions de dollars)
1	Néant.	0	0	0	0	0
2	Control Data 6 600-6 800. I.B.M. 360/92. Etc.	2	2	10	14	82
3	I.B.M. 7 090-9 411, 360/50-70. Control Data 3 400-3 800. Univac 1 107-8. Honeywell 1 800. Philco 2 000. Etc.	33	497	861	1 337,4	2 210,9
4	I.B.M. 704-709, 7 040-44, 360/30-40. Univac III. Honeywell 800, 1 400. C.D.C.-1 604, 924. A.S.I. 2 100, 420, 210. Etc.	30	776	2 779	1 889,1	1 407,7
5	I.B.M. 701-705, 7 070, 1 401, 1 460. Univac 1 004, 1 100 Séries. N.C.R. 315. G.E. 210-225, 415-435. S.D.S. 910-910. Etc.	30	11 742	4 132	3 152,4	1 304,5
6	I.B.M. 650, 1 410, 1 440, 1 620. R.C.A.-301, 501. Univac 80/90, 1 050, I-II. Burroughs-205-280. Honeywell Data 1 000, H-400. Etc.	23	5 144	2 131	1 432,4	440,0
7	N.C.R.-390. Monroe Monrobot XI. C.D.C.G.-15. General Precision L.G.P.-21-30. I.B.M. 305. Etc.	21	3 481	1 520	285,9	57,7

FIG. I. - Distribution des machines à calculer aux États-Unis en 1965

a) **La calculatrice « de bureau »**



b) **Implantation d'une calculatrice électronique**

FIG. 2. - Les deux étapes du calcul automatique

détaillée des calculatrices électroniques. Le lecteur qui serait curieux de tels détails pourrait se reporter avec fruit à la bibliographie.

Mais pour la compréhension de la suite, il suffira de garder présente à l'esprit l'analogie que développe la figure 2 :

L'opératrice mécanographe qui utilise une machine à calculer de bureau correspond au pupitre de commande et aux organes de lectures de la calculatrice électronique : elle lit les factures, etc., et en extrait les informations numériques qu'elle doit additionner. Dans le cas de la calculatrice électronique, ces informations initiales sont

stockées sous forme de cartes perforées, de bandes perforées ou d'enregistrements magnétiques et sont lues par des organes spéciaux. Il en va de même des informations finales, c'est-à-dire des résultats.
La machine à calculer de bureau elle-même correspond à l'unité centrale de la calculatrice électronique : c'est là que s'effectuent les opérations arithmétiques proprement dites. Dans le cas de la calculatrice électronique, on peut effectuer un très grand nombre d'opérations et pas seulement les quatre opérations arithmétiques (nous en verrons quelques exemples au chapitre suivant).
Enfin, et c'est le point capital, la calculatrice électronique dispose d'une *mémoire,* c'est-à-dire qu'elle peut stocker des informations intermédiaires, ce qui correspond, dans le cas du calcul de bureau à un stockage de feuilles ou de bandes de papier en vue de leur réutilisation ultérieure.
Du fait que la calculatrice électronique effectue près d'un million d'opérations par seconde, la vitesse de calcul est donc plus d'un million de fois supérieure à celle d'un calculateur humain, même aidé par une machine de bureau. En outre, le fait qu'elle transporte les informations, au même rythme d'un support matériel à un autre, empêche l'utilisateur de suivre et de contrôler lui-même le déroulement des opérations. Il lui faut donc rédiger à l'avance la suite des *instructions* qui devront être exécutées, et les transformer dans le langage qu'utilise la machine (par exemple des cartes perforées). Il faut enfin abandonner le contrôle du *programme* ainsi constitué à la machine elle-même.
L'originalité des machines à calculer réside donc dans le fait qu'elles permettent un automatisme appliqué non seulement aux *actions* élémentaires, mais aussi à leur *enchaînement.*

***

Si maintenant on compare les paragraphes que nous avons consacrés successivement au développement de l'intelligence et à celui des automatismes, notamment des machines à calculer, on voit apparaître quelques similitudes. Elles sont toutes fondées sur l'existence de techniques de *représentation* : objets extérieurs transformés en « informations» qui seront traitées par des organes adéquats. Cela n'implique,

nous le répétons, aucune identité avec les mécanismes du cerveau. Par contre, cela confirme clairement que les spécifications techniques propres au domaine de l'intelligence artificielle n'introduisent pas de restrictions de *principe.* Dans les chapitres suivants, nous montrerons sur des exemples que les problèmes appartenant aux régions les plus variées de l'intelligence sont *effectivement* pris en compte.
Cela implique surtout la possibilité d'exprimer les problèmes des deux domaines à l'aide d'un même *langage* et c'est ce que nous allons chercher à déterminer au cours du chapitre suivant.

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.

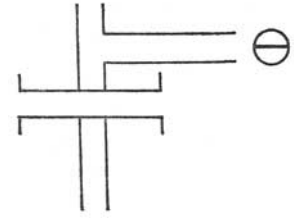


FIG. 3. - Exemple de message bidimensionnel

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).

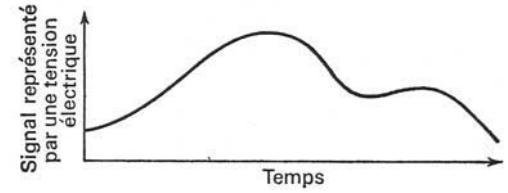


FIG. 4. - Représentation graphique d'un signal analogique élémentaire

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.

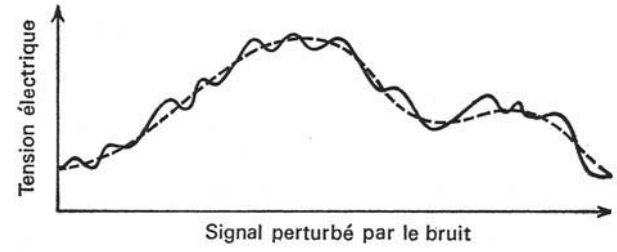
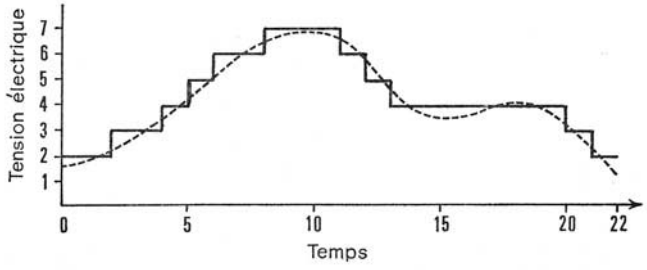


FIG. 5. - Echantillon d'un signal
qui sera identifié au signal indiqué en pointillé

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.

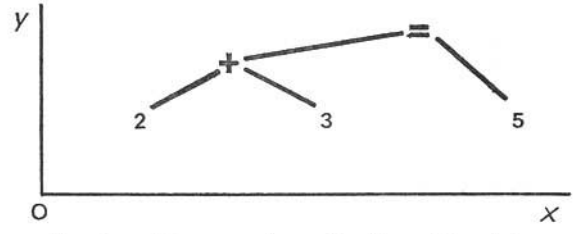


FIG. 8. — Une expression arithmétique élémentaire représentée à l'aide d'un « graphe »

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

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

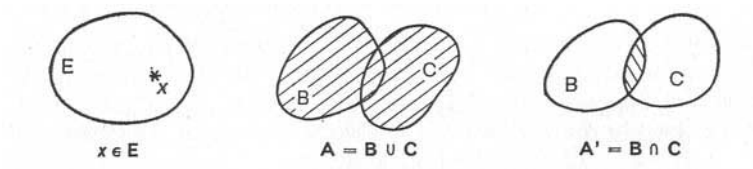


FIG 9 - Sémantique élémentaire de la théorie des ensembles

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

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

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

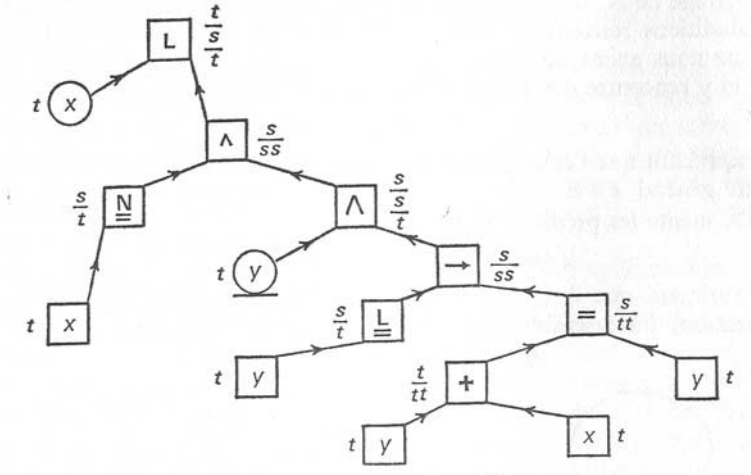


FIG. 10. — Représentation graphique d'une expression contenant des opérations et des prédicats arithmétiques et logiques

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.

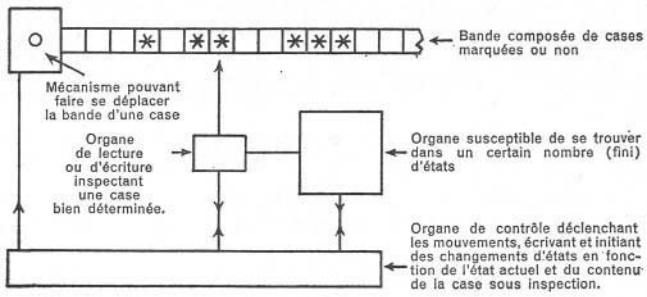


FIG. 11. — Représentation schématique d'une machine de Turing

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.*

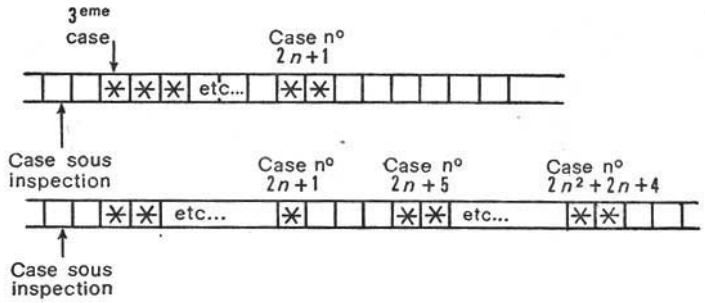


FIG 12 : état initial et état final de la bande
lors d'une élévation au carré dans une machine de Turing

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. I_1
2. I_2
3. I_3

avec

$I_1 = +2, t(2)$
$I_2 = e,$ RTZ, RTZ, $m,$ *LTZ, LTZ,* $m,$ $+2, t(2),$ –, LTZ, $+2, t(3)$
$I_3 = e, +,$ *LTZ,* $+2,$ $t(2)$

mais bien

I. I_1 (2)
2. I_2 (2, 3)
3. I_3 (2)

avec

$I_1(n) = +2, t(n)$
$I_2(n_1, n_2) = e,$ RTZ, RTZ, $m,$ LTZ, *LTZ,* $m,$ $+2,$ $t(n_1),$ –, LTZ, $+2, t(n_2)$
$I_3(n) = e, +,$ 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,

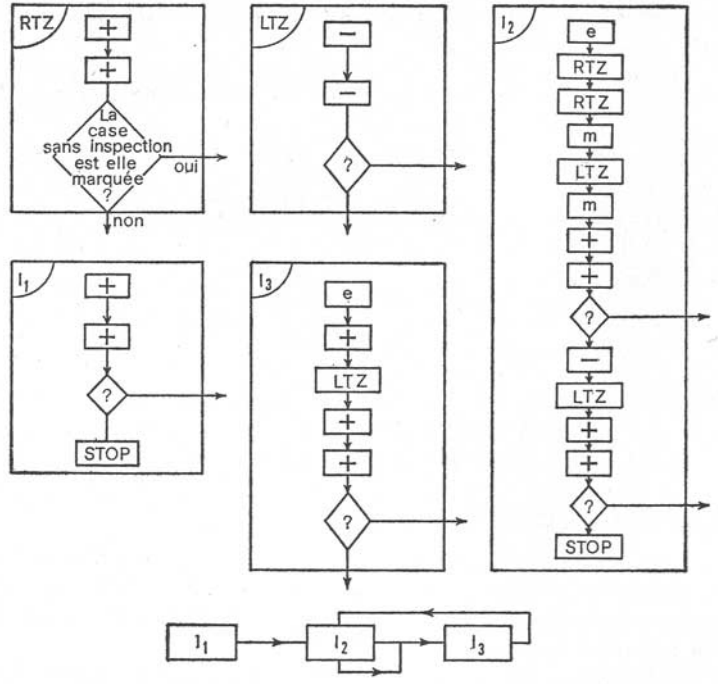


FIG. 13. - Schéma des macro-instructions et du programme principal
Les flèches indiquent l'enchaînement des instructions élémentaires
(ou des macro-instructions)

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

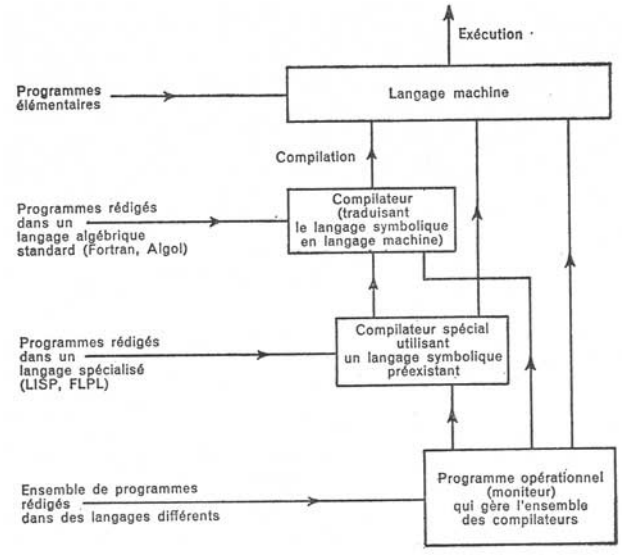


FIG. 14. – Hiérarchie des langages de programmation

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

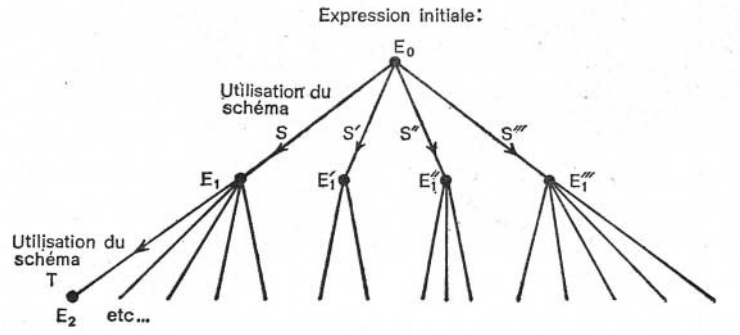
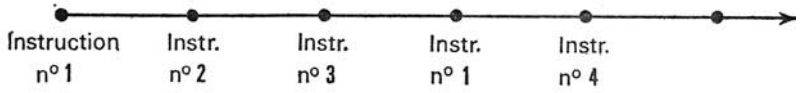


FIG. 55. - Développement « arborescent » d'expressions formelle.
Pour chaque expression, on a le *choix* entre un certain nombre de schémas qui lui sont applicables

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.

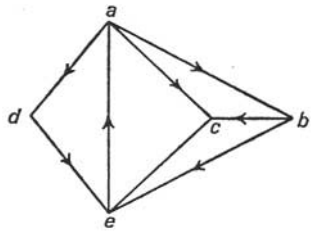


FIG. I6
Représentation « sagittale » d'un graphe

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,

les jeux - au moins ceux que nous considérerons ici - se définissent à l'aide d'un système de règles complètement explicitées. Mais leur conduite, contrairement aux mathématiques usuelles, se fait en énonçant complètement chaque pas, sans aucun raccourci, sous-entendu ou «abus de langage ».

Ou encore, si l'on considère une partie comme une phrase où les mots définissent les coups joués - on a donc ici un dialogue -, les règles de grammaire sont les règles du jeu et le point final le constat du gain ou de l'échec ; on voit qu'on dispose là d'une langue fort simple malgré la grande variété de ce qu'elle permet de dire !

Pour illustrer cette remarque nous allons en développer les conséquences dans le cas d'un jeu de cartes particulièrement simple.

Imaginons un jeu de trente-deux cartes usuel. Pour simplifier nous dénoterons les couleurs par les signes K (pour carreau), Q (pour coeur), P (pour pique), T (pour trèfle) et les valeurs par R (roi), S (reine), V (valet), I0, 9, 8 et 7. Les joueurs seront désignés respectivement par A et B. Supposons que chaque joueur ait reçu sept cartes et qu'on ait retourné une des cartes restantes pour désigner l'atout.

Dès lors une partie pourra être considérée comme un texte composé de sept « paragraphes ». Chaque paragraphe comprend trois « phrases » ; la première phrase comprend trois « mots » : un symbole de couleur, un symbole de valeur et un nom de joueur. La phrase suivante est de même nature, mais le nom du joueur doit être différent. Enfin la troisième phrase ne comprend qu'un seul mot : le nom du joueur qui emporte le pli. Dès lors le nom du joueur figurant dans la première phrase du paragraphe suivant devra être le même.

Ces règles assurent ce qu'on pourrait appeler la cohérence *syntaxique* du texte.

Il semble naturel de rattacher au niveau *sémantique* la règle qui désigne le joueur emportant le pli en tenant compte d'une relation d'ordre entre valeurs à l'intérieur d'une même couleur, et de la priorité accordée soit à l'atout, soit, en l'absence d'atout, au joueur ayant déposé la première carte dans le pli.

Enfin le calcul des plis à la fin de la partie, en vue de la proclamation du vainqueur, appartient évidemment au niveau *pragmatique.*

Bien entendu les « textes » ainsi rédigés semblent fort différents de notre langage naturel, mais au fond certaines formes de poésies le

sont également. Par contre syntaxe et sémantique sont ici d'une définition autrement aisée que dans le cas du langage naturel.

L'image linguistique est donc particulièrement commode pour exprimer la structure du jeu, notamment en ce qui concerne l'énoncé des règles.

Par contre la découverte d'une *stratégie* propre à assurer le succès, c'est-à-dire le gain de la partie, si elle peut être aidée par l'utilisation d'un modèle linguistique, c'est-à-dire d'une formalisation, nous oblige à mettre en oeuvre d'autres concepts, situés à un niveau différent mais qui, bien entendu, sont, eux aussi, susceptibles d'être formalisés.

C'est ainsi que l'une des notions qui se présentent le plus naturellement à l'esprit d'un joueur est celle de la *distance* qui le sépare encore de la victoire.

Dans les jeux où l'habileté est essentiellement liée à un effort physique, cette distance est facile à définir : distance (au sens géométrique) restant à parcourir, écart à combler entre les buts marqués, etc.

Dans les jeux « intellectuels », les plus élémentaires, la définition de la distance reste transparente : nombre de dominos restant à caser, nombre de pièces d'un puzzle restant à assembler, distance restant à parcourir sur le « chemin » du jeu de l'oie, etc.

Pour les jeux intellectuels plus compliqués comme le jeu des échecs, il n'existe pas de définition naturelle de la distance, et il en va de même pour des tâches intelligentes que nous examinerons par la suite : démonstration des théorèmes, traduction automatique.

Définir une distance entre « situations » au cours d'un jeu et, grâce à cette définition, déterminer les conditions d'un chemin optimal (ici, un chemin conduisant au gain) est donc une contribution importante à l'ensemble des rubriques de l'intelligence artificielle. On peut même dire que c'est là l'essence de cette « heuristique », notion mal définie mais très suggestive, que l'on évoque bien souvent dans les tentatives qui appartiennent à notre domaine.

En résumé, il existe deux voies d'accès principales à la simulation des jeux intellectuels : un point de vue « géométrique », où la notion de « situation » et de « distance » joue un rôle essentiel, et un point de vue « linguistique » où l'on met l'accent sur l'énoncé des situations et des actions. L'automatisation établit naturellement un lien entre ces deux points de vue puisque le cheminement dans l'espace des situations

est accompli par des actions dont l'énoncé définit une macro-instruction, au sens que nous avons défini dans le chapitre précédent.
Les jeux les plus simples (cache-tampon, quatre coins, etc.) font appel souvent à des éléments géométriques et physiques de l'univers extérieur que les automates actuels, meubles rigides et fixes, ne peuvent évidemment pas atteindre directement. Certes, une représentation symbolique des lieux et des déplacements n'est nullement exclue, mais nous nous limitons pour le moment aux jeux dont la définition même ne comporte que l'échange d'informations, mots, lettres, chiffres ou autres symboles aisément codifiables.
Enfin une distinction s'impose entre les jeux qui font intervenir un élément de hasard (jeux de cartes en particulier) et ceux dont le développement est entièrement déterminé par la seule initiative des joueurs (le jeu des échecs est l'exemple le plus couramment utilisé).
Les deux types de jeux peuvent être abordés par les automates, mais il est clair que l'intervention du hasard n'apporte rien de significatif. Aussi nous concentrerons-nous sur l'étude de jeux appartenant au deuxième type, en remarquant toutefois qu'il existe des programmes ou des fragments de programmes relatifs aux jeux de cartes et notamment au bridge.
Les jeux dans lesquels le hasard n'intervient pas sont appelés « jeux à information complète ».

ELEMENTS D'UNE THEORIE DES JEUX

Si nous avons invoqué, au début de ce chapitre sur les jeux, l'importance qu'ils possèdent dans l'apprentissage de l'intelligence naturelle, nous n'avons pas l'intention d'aborder leur simulation par les machines à calculer sous l'angle de l'apprentissage. Certes il y a eu - et il y a encore - de nombreux travaux relatifs à la simulation de l'apprentissage par les machines. Ces travaux sont même rangés par certains auteurs (notamment par Minsky) dans le cadre de l'intelligence artificielle.
Il nous semble que les jeux ne sont simulés de façon intéressante que si l'on introduit d'emblée dans le programme une information théorique préalable (nous renonçons donc délibérément, encore une fois, au point de vue anthropomorphique).
C'est cette information théorique que nous allons présenter maintenant.

Pour essayer d'analyser, au point de vue théorique, ce qui se produit au cours d'un jeu « à deux personnes » (personnes que nous appellerons respectivement « blanc » et « noir »), nous devons tout d'abord définir une « position». On appellera « position» le couple constitué par un *diagramme* et par le *trait.* Le diagramme sera la place des pièces sur l'échiquier, des pions sur le damier, etc., à un instant donné. Le trait désigne celui des joueurs qui doit jouer à cet instant.
On a donc deux ensembles : E_b = ensemble de toutes les positions possibles avec trait aux « blancs » ; E_n = ensemble de toutes les positions possibles avec trait aux « noirs ». Évidemment, le trait passe alternativement d'un joueur à l'autre.
En utilisant les notations ensemblistes classiques (que nous avons rappelées au chapitre précédent), on pourra formaliser l'algèbre des situations et la logique du succès et de l'échec (I).

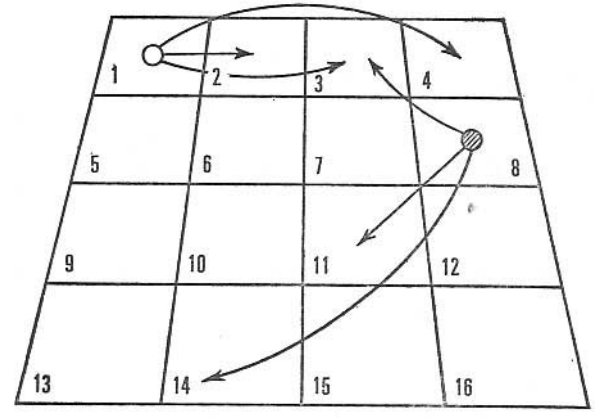


FIG. 17. - Représentation géométrique d'un jeu à deux personnes

Pour fixer les idées, imaginons une représentation géométrique du jeu comme celle de la figure t7.

(I) Pour ce paragraphe on se reportera avec fruit au petit livre de C. BERGE, *Théorie générale des jeux* à n *personnes,* Gauthier-Villars, 1960.

Les flèches représentent les mouvements possibles du pion blanc et du pion noir. On suppose qu'un pion n'a pas le droit d'en chevaucher ou d'en sauter un autre.
A la position initiale on peut associer l'expression :

$$x = \{ (b,I) ; (n,8) \}$$

puis les positions possibles suivantes :

$$y_1 = \{ (b,2) ; (n,8) \}$$
$$y_2 = \{ (b,3) ; (n,8) \}$$
$$y_3 = \{ (b,4) ; (n,8) \}, \text{ etc.}$$

ce qui se représente aisément par le graphe de la figure 18.

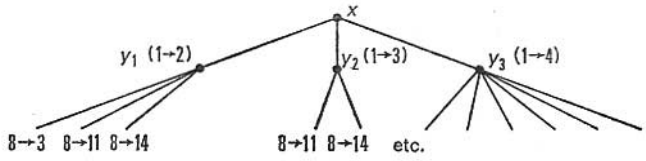


Fig. 18. - Développement arborescent des coups possibles à partir de la position représentée par la figure 17

Une position x étant donnée, on désigne par Γx l'ensemble de toutes les positions qui sont des *successeurs* possibles pour x ; la loi Γ est une « application multivoque » :

$$\text{si } x \in E_b \qquad \text{on a } \Gamma x \subset E_n$$
$$\text{si } x \in E_n \qquad \text{on a } \Gamma x \subset E_b .$$

Une *position de gain aux blancs* est une position $z \in E_b$ telle que $\Gamma z = \emptyset$. Leur ensemble est $M_b \subset E_b$.
On définira de même *l'ensemble de gain aux noirs* M, et l'ensemble de jeu nul $P = \{ z / \Gamma z = \emptyset \} - (M_b \cup M_n)$.
Le jeu se termine si la position $x \in P \cup M_b \cup M_n$, mais dans ce modèle, il peut aussi ne jamais se terminer.
La théorie mathématique du jeu se divise alors en deux : une théorie locale et une théorie globale.

I° La théorie locale repose sur l'idée de *tactique :* à toute position $x \in E_n$, les blancs attachent une valeur numérique $f(x)$ qui est d'autant plus grande que la position x est jugée bonne pour les blancs. f est la *fonction de préférence.* On prendra :

$$0 \leq f(x) \leq I$$
$$f(x) = I$$

et on peut définir :

- une tactique d'ordre I - si $x \in E_b$, les blancs choisissent dans Γx la position y telle que :

$$f(y) = \max_{z \in \Gamma x} f(z) ;$$

- une tactique d'ordre 2 - si $x \in E_b$, et si au coup suivant la position est $x' \in E_b$, les blancs peuvent garantir :

$$\max_{y' \in \Gamma x'} f(y').$$

Si les blancs choisissent dans Γx une position $y \in E_n$, les noirs vont choisir dans Γy une position x' telle que l'expression ci-dessus soit aussi petite que possible ; ils peuvent garantir :

$$\min_{x' \in \Gamma y} \max_{y' \in \Gamma x'} f(y') = f^{(1)}(y).$$

Une tactique d'ordre 2 consistera pour les blancs à choisir dans Px une position y telle que :

$$f^{(1)}(y) = \max_{z \in \Gamma x} f^{(1)}(z).$$

La théorie locale est donc relative au choix de la fonction de préférence f. Or, dans le cas général, il n'existe pas de déduction simple de f à partir des règles du jeu. Le choix de f résulte d'un ensemble d'intuition et de tâtonnements. Il est « heuristique » !
2° La théorie globale repose sur l'idée de stratégie ; pour les blancs, *la stratégie,* d'après von Neumann, est une loi qui fait correspondre à toute position $x \in E_b$ une position $y \in \Gamma x$ que l'on note $y = \sigma(x)$. σ est alors une application univoque.

On peut donner une description ensembliste de l'ensemble des positions gagnantes.
Si A c E_b par exemple, on pose :

$$\bar{\Gamma}A = \{\ x/\Gamma\, x \cap A \neq \varnothing\ \}$$
$$\text{et}\quad \overset{+}{\Gamma}A = \{\ x/\Gamma\, x \neq \varnothing\ \Gamma\, x \subset A\ \}.$$

L'ensemble des positions gagnantes d'ordre i pour les blancs peut donc s'écrire :

$$G\,(I) = \bar{\Gamma}\, M_b.$$

L'ensemble des positions qui donnent le gain en deux coups, ou position gagnante d'ordre *2* pour les blancs, peut donc s'écrire :

$$G\,(2) = G\,(I) \cup \bar{\Gamma}\, \overset{+}{\Gamma}\, G\,(I).$$

Cette méthode consiste, en somme, à supposer le problème résolu on se donne les positions gagnantes possibles, puis celles qui, compte tenu des règles, pouvaient les précéder, et on remonte en arrière.
En général, l'ensemble des positions qui amènent à la victoire en *n* coups peut se définir par l'expression :

$$G\,(n) = G\,(n\text{-}1) \cup \bar{\Gamma}\, \overset{+}{\Gamma}\, G\,(n\text{-}1).$$

ou plus symboliquement encore :

$$G\,(n) = (I \cup \bar{\Gamma}\, \overset{+}{\Gamma})\, G\,(n\text{-}1).$$

Par induction, on voit donc que l'ensemble des positions qui donnent le gain en *n* coups est :

$$G\,(n) = (I \cup \bar{\Gamma}\, \overset{+}{\Gamma})^{(n-1)}\, \bar{\Gamma}\, M_b.$$

Si l'on suppose que le jeu a une durée limitée, l'ensemble des positions gagnantes sera la réunion de tous les $G\,(n)$ (pour $n = 1, 2, \ldots$).
(En réalité la formule correcte pour l'ensemble des positions gagnantes est la réunion de tous les G (α) pour α = nombre ordinal transfini,

mais nous n'entrerons pas ici dans une analyse qui deviendrait vite trop abstraite.)
De cette formule qui, elle, est rigoureuse, et ne dépend d'aucune fonction f à déterminer intuitivement, on ne peut pas, malheureusement, déduire une construction *effective* de l'ensemble des positions gagnantes. par contre, dans le cas du jeu des échecs, on en déduit aisément des théorèmes, par exemple le théorème de Zermelo-Kalmar-von Neumann, qui s'énonce : « Ou bien il existe une stratégie gagnante pour les blancs, ou bien il existe une stratégie gagnante pour les noirs, ou bien chacun des deux joueurs a une stratégie qui lui garantit la nullité. »
Ce théorème n'est pas trivial comme il semble à première vue, car il suffit de changer très légèrement la règle du jeu des échecs pour que l'on ait la situation suivante : les blancs peuvent garantir la nullité par une stratégie mais les noirs ne peuvent rien garantir du tout à l'aide d'une stratégie.

ESSAI DE PROGRAMMATION D'UN AUTOMATE POUR SIMULER UN JOUEUR

Les théorèmes du type de ceux que nous venons de rencontrer sont des théorèmes d'existence. Avoir démontré qu'il existe une stratégie gagnante peut satisfaire le mathématicien - ou le philosophe -, mais sûrement pas le joueur. Cela ne satisfait pas non plus celui qui veut simuler un joueur à l'aide d'une machine à calculer, car sa préoccupation est d'amener l'automate à développer un jeu correct et, si possible, gagnant, mais le gain n'est pas vraiment essentiel.
La nécessité de préparer un programme réel qui permette à l'automate de jouer, et de jouer aussi bien que possible, permet de donner un éclairage nouveau à la distinction présentée ci-dessus entre théorie locale et théorie globale. Mais il nous faut d'abord préciser en quels termes nous allons décrire ce programme.
Soit B et N les camps en présence. Nommons b_i et n_j les coups joués respectivement par B et N ; une partie sera équivalente à une suite $b_1, n_1; b_2, n_2; \ldots x_p$.
Suivant que x_p est un b ou un n, B ou N aura gagné la partie en *p* coups.

Supposons que le joueur N soit remplacé par un automate, le programme d'ensemble pour le jeu sera le résultat d'une combinaison de programmes partiels que nous allons décrire brièvement ci-dessous.

L'organisation générale de ces programmes partiels est indiquée dans le schéma de la figure 19.

Pour décrire chaque sous-programme nous utiliserons un format standard donnant le nom du sous-programme et son abréviation, puis, sous la rubrique « algorithme », l'énoncé de ce qu'il doit faire ; sous la rubrique « documentation », l'ensemble des informations permanentes qu'il doit contenir ; sous la rubrique « données » les informa-

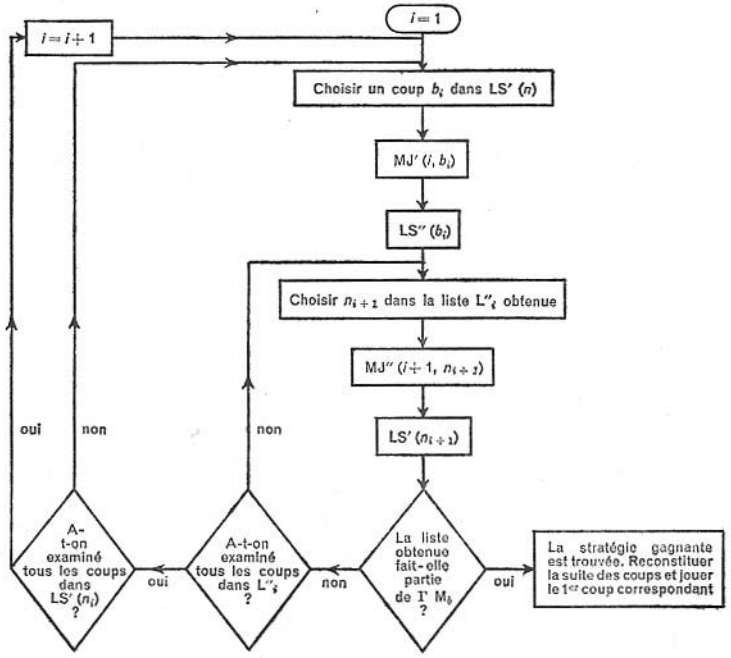


FIG. 19. - Schéma de principe du programme de simulation d'un joueur

tions qui se renouvellent à chaque exécution du programme ; et enfin sous la rubrique « résultat », ce qui sort de l'automate lorsque le programme est exécuté.

On peut ainsi définir les programmes suivants :

PJ
Programme général de jeu

algorithme : choix du coup n_i ;
documentation de base : règles du jeu + tactique ;
données : coup de l'adversaire b_i + situation du jeu au temps i - 1 ;
résultat : symbole représentant le coup n_i.

Le programme PJ se construira à l'aide des sous-programmes suivants :

MJ' (*i*, b_i)
Mise à jour de la situation au temps *i*

algorithme : transformation des tableaux de situation à la suite du coup b_i ;
documentation de base : définition des tableaux ;
données : T''_{i-1} et b_i ;
résultat : T' (1).

On construira symétriquement le programme MJ" *(i, n_i).*

LS" (b_i)
Recherche de l'ensemble des coups possibles par N après b_i

algorithme : examen successif des « pièces » du coup N ;
documentation de base : règles du jeu;
données : T'_i ;
résultat : L''_i .

On construira symétriquement le programme LS' (*n*).

On pourrait, en principe, à l'aide de ces sous-programmes, effectuer le choix du coup à jouer, au sein de la liste L''_i fournie par le couple MJ', LS" à chaque pas.

PREMIERE RENCONTRE AVEC LE « MUR DU COMBINATOIRE » ·

Supposons - ce qui n'est jamais vrai, mais qui nous permettra une estimation grossière des ordres de grandeur en présence - que chaque joueur ait, à chaque coup, le choix entre 10 coups possibles.

(1) Les T'_i et T''_j représentent l'état du jeu pour les jours B et N aux temps *i* et *j*.

Il faudra donc explorer au premier niveau io possibilités, au second niveau $10^2 = 100$ possibilités, etc.
Au ne niveau il faudra explorer 10^n possibilités.
Si la vitesse et la capacité de mémoire de l'automate sont illimitées, on peut de la sorte atteindre les positions gagnantes de N de toutes les façons possibles. Et, au fond, le théorème de Zermelo-von Neumann, dont nous avons présenté plus haut une variante, ne fait qu'exprimer cette possibilité.
D'après le théorème, on sait qu'à tout moment il est *théoriquement* possible de désigner le vainqueur. Par contre, on ne peut pas, en général, décrire explicitement les démarches qui entraînent cette victoire. C'est là une situation bien connue en mathématique où l'on distingue les définitions existentielles et les définitions constructivistes.
C'est ainsi que dans le cas des jeux très simples, comme le jeu de Nim ou le Tic-Tac-Dou, il est possible d'expliciter la méthode victorieuse (c'est l'algorithme du jeu), alors que pour un jeu complexe comme celui des échecs on ne possède pas d'algorithme. Dans ce cas, il faut développer des méthodes d'approche que l'on nomme «heuristiques », et qui, sans assurer la victoire, la rendent relativement probable.
Ce qui distingue au premier coup d'oeil un jeu comme le Nim (jeu rendu célèbre par un épisode du film *L'année dernière à Marienbad),* des dames ou des échecs, c'est évidemment le nombre des «pièces » en présence. Par la même occasion le nombre maximum de coups possibles est diminué et ceci de façon exponentielle.
Cette caractéristique de tels problèmes, c'est-à-dire cette dépendance vis-à-vis de fonctions de type exponentiel, conduisant, lors de certaines évaluations, à des chiffres astronomiques, c'est celle de tous les problèmes *combinatoires.* Le problème des jeux est l'un d'entre eux.
Une première conclusion se dégage cependant de ce qui précède même pour la démonstration de théorèmes généraux relatifs aux jeux, il n'est pas inutile d'utiliser la notion d'automate, en exploitant évidemment un modèle purement théorique d'automate, dont la capacité de mémoire est illimitée et dont la vitesse peut également être rendue aussi grande que l'on veut.
Mais, bien entendu, pour tous les jeux présentant quelque intérêt, il n'existe pas d'automate répondant à de telles exigences. Il faut donc, pour sélectionner le meilleur coup, faire appel à des critères moins

évidents et moins généraux. Chaque jeu pose ainsi un problème nouveau, une fonction *f* nouvelle à choisir.
Cette fonction *f* permet, à chaque fois, de remplacer l'examen de tous les coups suivants possibles par celui d'un seul coup (ou en tout cas d'un très petit nombre de coups). Dans ces conditions, l'examen de toutes les ramifications est considérablement réduit.
Dans certains cas la réduction est telle (ou encore le pouvoir *anti-combinatoire* de la fonction de préférence est tel) que l'examen de l'arborescence peut être effectivement accompli par un automate.
C'est ce qu'on verra mieux en examinant en détail un jeu particulier, ou plutôt une classe de jeux voisins les uns des autres. Nous pourrons alors détailler les programmes MJ et LS décrits plus haut, et définir des programmes de sélection plus fins. Pour un cas particulier, la programmation a été réellement effectuée et l'expérience conduite sur machine. Elle semble instructive et permet de dégager des perspectives générales qui s'offrent au développement de la simulation des jeux par les machines.

ÉTUDE D'UN EXEMPLE

Les jeux que nous utiliserons se rangent dans la classe de ce que nous proposons de nommer « les jeux de grille ».
Chaque jeu appartenant à la classe est caractérisé par deux nombres entiers a et b. Chaque jeu particulier sera appelé « le jeu de grille *(a, b) »* où a et b sont fixés. Le premier entier a détermine les dimensions de la grille (il s'agit donc d'une grille carrée à a lignes et à a colonnes). Chaque joueur dispose à son tour un pion à son emblème au croisement d'une ligne et d'une colonne. Le diagramme est ici l'image de la grille et des pions disposés à ses nceuds. Le vainqueur est celui qui réussit à disposer b de ses propres pions en alignement.
On doit avoir évidemment $a \leq b$.
Si $a = b = 3$, il s'agit du Tic-Tac-Dou.
Si $a = 19$, $b = 5$, il s'agit du Go-Bang.
Si a est très grand, $b = 5$, il s'agit du « Morpion » bien connu des lycéens.

Remarquons d'abord que si $a = b = 2$, le jeu n'a aucun intérêt car, quelle que soit la tactique adoptée, le premier joueur gagne.

Pour $a = b = 3$, une règle simple permet d'obtenir automatiquement le succès, ou au moins le jeu nul pour le premier joueur.

Lorsque $b > 3$, il faut que a devienne beaucoup plus grand que b pour que le jeu présente un intérêt. La raison en sera analysée en détail au chapitre VI relatif à la notion de « complexité ».

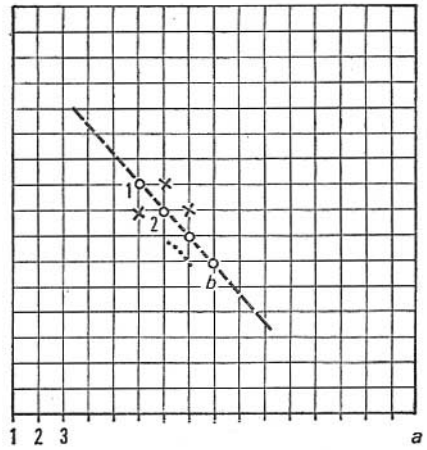


FIG. 20. — Exemple d'un jeu de grille de paramètres a et b

Restant pour le moment au niveau du jeu, nous nous limiterons à étudier la programmation d'un automate qui, placé devant une situation donnée de la grille, doit choisir le coup suivant. Il n'y aurait évidemment aucune difficulté de principe à traiter le problème d'un jeu complet.

Nous décomposerons ce programme en un ensemble de programmes simples, aux fonctions bien déterminées, ensemble dont l'articulation sera précisée par la suite.

Nous allons surtout insister sur le traitement de ce que nous avons appelé un peu plus haut les données, c'est-à-dire les informations relatives à la situation des pions déposés par les joueurs sur la grille.

Le programme correspondant est le suivant :

LG Lecture de la grille
algorithme : une liste de a^2 registres est associée à la grille ; chaque registre reçoit le contenu 0, 1 ou 2, suivant que le noeud de la grille est inoccupé, a reçu un pion du joueur A ou du joueur B ;
documentation de base : état de la grille au coup précédent;
données : deux coups joués par l'adversaire ;
résultat : grille mise à jour.

Ce programme donne en quelque sorte à l'automate la perception de la grille. Il faut maintenant pouvoir introduire quelques « concepts » rudimentaires. Ceci se fait au niveau du programme.

Mais pour que le programme soit efficace, il est important de choisir avec soin le format sous lequel les données y sont introduites.

Dans l'exemple actuel, la «vision» que l'automate peut avoir du jeu se situe sur un plan *statique :* la vision du diagramme lui-même, et sur un plan *dynamnique,* la vision des combinaisons déjà réalisées (2, 3 pions déjà alignés, etc.).

a) Nous appellerons TJ un tableau de 361 cases qui sont des « mots » de 36 bits, c'est-à-dire des suites de 36 chiffres formées de o et de i uniquement.

2 bits définissent l'occupation de la case

- 00 : la case est libre ;
- 01 : la case est occupée par un pion de la machine;
- 10 : la case est occupée par un pion de l'adversaire ;
- 11 : une erreur a été commise dans la construction du tableau.

Les 34 bits restants comprennent

- 17 bits exprimant la grandeur de la fonction d'évaluation relative à la machine en ce point ;
- 17 bits exprimant la grandeur de la fonction d'évaluation relative à l'adversaire en ce point.

b) Nous appellerons *n*-uplet un ensemble de n points alignés et occupés par un même joueur avec la condition que l'adversaire ne possède aucun point dans ce domaine ainsi défini, et que, d'autre

part, ce domaine n'excède pas 5 cases. Il peut y avoir plusieurs types de *n*-uplets et ils peuvent posséder différentes orientations. On a les types suivants :

singulets : ce sont les pions isolés déposés par chaque joueur ;
doublets : ce sont les couples de pions déposés de la façon suivante :

```
X X . . .      X . X . .      X . . X .
X . . . X
```

triplets :

```
X X X . .      X X . X .      X X . . X
X . X . X
```

quadruplets :

```
X X X X .      X X X . X      X X . X X
```

quintuplets :

```
X X X X X
```

... et les orientations suivantes :

H : horizontal ;
D : diagonale montant vers la droite ;
V : vertical ;
G : diagonale montant vers la gauche.

Compte tenu de ces conventions, il faut s'efforcer *d'organiser* les données de façon à encombrer le moins possible la mémoire de l'automate tout en conservant un accès aussi rapide que possible.

Il existe une organisation de ce genre qui est particulièrement commode, c'est l'organisation en *listes.* Les listes sont construites de la façon suivante : chaque information est divisée en deux parties la première est un nom (ou un titre), la seconde une adresse (le pointeur) ou une donnée (ici une position dans le tableau TJ). Lorsqu'on a ensemble de telles informations on peut en déduire une représentation graphique naturelle en reliant par une flèche la partie « adresse » d'une information à l'information contenue dans cette adresse. Sans entrer dans les détails de cette organisation nous en présentons, dans la figure 21, deux applications particulières :

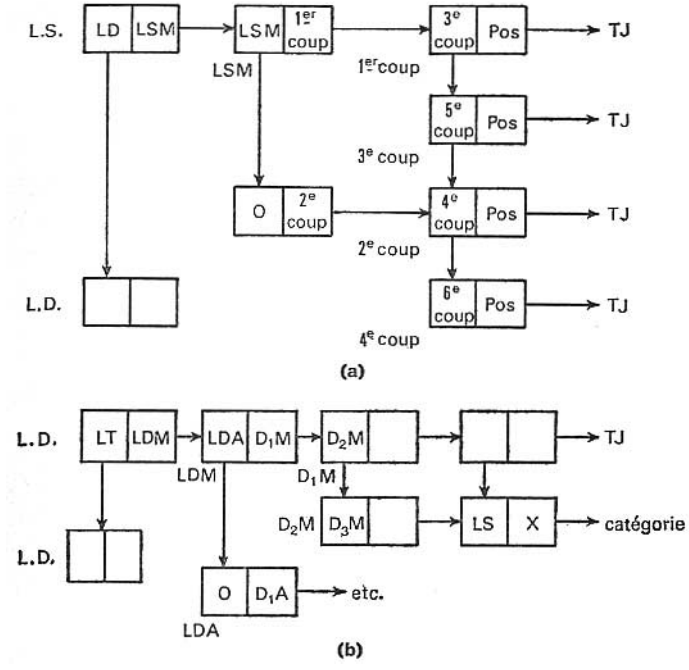


FIG. 21. - Modèle d'organisation en « liste » des singulets et des doublets

L'indication « catégorie » est une information de 9 bits qui spécifie d'une part le *type* topologique du doublet, d'autre part l'*orientation* de ce doublet. 5 bits sont utilisés pour le type :

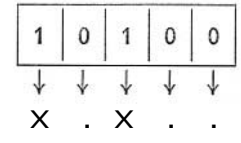


FIG. 22. — Correspondance entre l'arrangement topologique d'un *n*-uplet et le symbole binaire de son « type »

Enfin 2 bits expriment l'orientation en *n*-uplet :

- 00 : horizontal (H) ;
- 01 : diagonale montant vers la droite (DD) ;
- 10 : vertical (V) ;
- 11 : diagonale montant vers la gauche (DG).

Enfin, 2 bits dénotent la fonction de fermeture, c'est-à-dire l'occupation des positions voisines du doublet par l'adversaire :

- 00 : libre aux deux extrémités ;
- 01 : fermé à droite ;
- 10 : fermé à gauche ;
- 11 : fermé aux deux extrémités.

Les autres listes (des triplets, quadruplets, quintuplets) sont évidemment structurées de la même façon.

c) Calcul des listes. — Chaque fois qu'un joueur annonce le coup qu'il joue, la machine doit mettre à jour le tableau et les listes. Pour cela elle dispose d'un tableau spécial TOE des opérations élémentaires nécessaires au balayage d'une file de 5 points alignés. Ces instructions sont les suivantes

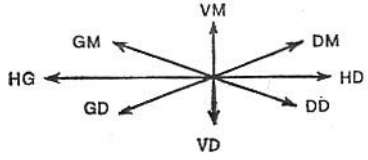


Fig. 23. - Les divers déplacements susceptibles d'assurer le balayage de la grille.

— o(1)	HD	ajouter + 1 à la 1re	coordonnée et	0 à la deuxième	
— 0(5)	HG	— — 1	—	0	—
— 0(2)	DM	— + 1	—	+ 1	—
— 0(6)	GD	— — 1	—	– 1	—
— 0(3)	VM	— 0	—	+ 1	—
— 0(7)	VD	— 0	—	— 1	—
— 0(4)	GM	— — 1	—	+ 1	—
— 0(8)	DD	— + 1	—	– 1	—

Ceci étant posé, le programme se construira comme suit :

1) on inscrit dans le tableau TJ à l'endroit où le coup vient d'être joué l'information relative à la machine du nouvel occupant ;
2) on ajoute à la liste LS l'adresse de cette case dans TJ ;
3) on complète les listes LD, LT, etc.

Nous allons maintenant pouvoir aborder, sur cet exemple, la recherche et le calcul d'une fonction d'évaluation.

a) Principe. — Dans un jeu relativement simple comme le Go-Bang, il est possible de faire jouer à la fonction d'évaluation un double rôle

- un rôle *stratégique* en déterminant les positions dominantes (au sens de la théorie des jeux) ;
- un rôle *tactique* en contribuant à la définition d'un minimax, lorsque aucune position dominante ne peut être atteinte.

C'est pourquoi les fonctions que nous avons considérées sont basées sur la combinaison de plusieurs informations :

- un tableau de valeurs associé à chacune des configurations de 1, 2, 3, 4, ou 5 pions identiques sur un même alignement de 5 cases ;
- un ou plusieurs filtres qui limitent le développement de l'arborescence aux branches conduisant à certaines de ces configurations;
- une règle d'arrêt qui limite la profondeur de l'arborescence à un nombre de coups donnés ;
- une formule qui donne l'évaluation pour la position en fonction du nombre des configurations obtenues aux extrémités de l'arborescence, de leurs valeurs et de la longueur de la branche qui la relie au sommet.

On a évidemment une certaine souplesse dans la construction de la fonction d'évaluation. Il convient de noter cependant que si l'on utilise cette fonction pour l'élaboration d'une stratégie de minimax il ne serait pas raisonnable que la fonction elle-même réclame, pour être calculée, l'examen d'une arborescence.

C'est pourquoi les deux premières fonctions que nous avons essayées correspondent respectivement à l'examen d'un ou de deux coups en avant. Une fois la valuation obtenue, il y a alors encore un sens à l'utiliser pour une exploration en profondeur et une stratégie minimax.

b) Description de notre première fonction. — Supposons que le jeu soit arrivé à un certain stade, on va alors construire les deux fonctions

$$f_{\text{ami}} \text{ et } f_{\text{ennemi}}$$

Pour les points où l'on a déjà joué

$$f_{\text{ami}} = f_{\text{ennemi}} = 0$$

Pour les autres, on calculera le poids des configurations nouvelles créées respectivement par le dépôt d'un pion ami ou ennemi (fig. 24).

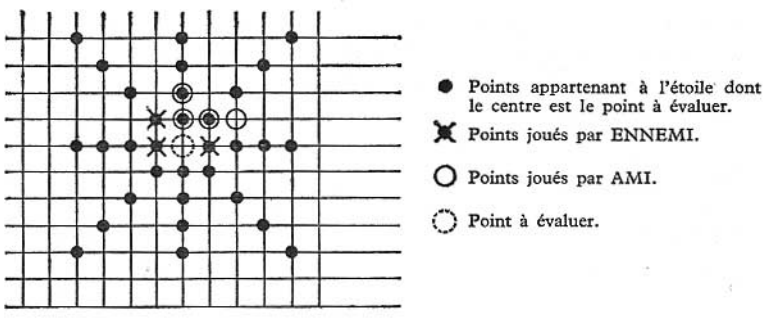


FIG. 24. — Utilisation d'une "étoile virtuelle" lors de l'évaluation d'une position

Le poids d'une configuration doit être tel que 5 pions alignés donnent la victoire et que l'absence de pions entraîne l'absence de valuation.

Une solution simple consiste à attribuer à un n-uplet $(n < 5)$ la valeur

$$k \frac{n}{5-n}$$

Lors de l'implantation du programme sur une calculatrice, les mémoires contiennent

– un tableau du jeu, entouré de ≠ dans lequel s'inscriront tous les coups ;

- une table des évaluations contenant
  a) le nom décodé de chaque case (exemple : la case B 07 s'écrit 427077 ; B = 42, 0 = 70, 7 = 77)
  b) l'adresse correspondante du tableau du jeu ;
  c) l'évaluation AMI pour la case correspondante ;
  d) l'évaluation ENNEMI pour la case correspondante ;
  au départ les évaluations sont toutes nulles ; de même dès qu'une case est occupée, ces évaluations deviennent nulles ;
- une table d'équivalence entre la situation du tableau et l'évaluation : par exemple si 5 cases consécutives d'une radiale ou d'une diagonale contiennent 10110, le tableau d'équivalence donne l'évaluation correspondante ;
- une zone de comparaison dans laquelle s'inscrit le contenu des 9 cases d'une diagonale, la position centrale étant celle de la case dont on calcule l'évaluation ;
- une table des coups joués, contenant, dans l'ordre, l'adresse des cases qui ont été occupées successivement.

Après chaque coup, seules les cases vides se trouvant sur l'étoile d'un diamètre de 9 cases (le centre de l'étoile étant la case du coup joué) seront réévaluées.

Pour chaque case à réévaluer, l'évaluation est d'abord remise à o. Ensuite chaque diagonale de 9 cases est mise dans la zone de comparaison.

Calcul de l'évaluation AMI. - Dans cette zone de comparaison, on examine chacun des 4 groupes de 5 cases successives. Si une de ces 5 cases contient un ≠ ou un coup ENNEMI, son évaluation est nulle. Sinon le dessin est comparé au tableau d'équivalence et on obtient ainsi l'évaluation pour ce groupe. Cette évaluation est ajoutée (!!!) à l'évaluation AMI de la case.

Calcul de l'évaluation ENNEMI. - Dans la zone de comparaison i est remplacé par i et réciproquement. On calcule alors comme pour l'évaluation AMI.

Lorsque des coups sont annulés, la machine fait successivement et dans l'ordre inverse des coups précédemment joués

1) remise à zéro de la case correspondante dans le tableau du jeu;
2) réévaluation de toutes les cases o se trouvant sur l'étoile.

Il peut être intéressant de donner quelques résultats expérimentaux relatifs à ce cas particulier.

Nous disposions tout d'abord d'un programme de calcul de fonction d'évaluation sur I.B.M.-I620; il s'agit d'un programme comportant 8oo instructions. Le calcul d'une nouvelle valuation, après qu'un coup vient d'être joué, prend en moyenne 50 secondes.

Nous avons tout d'abord effectué une série d'expériences semi-automatisées dans lesquelles la machine donne les deux évaluations AMI et ENNEMI de toutes les cases non occupées. Un opérateur humain utilise ces évaluations selon une tactique déterminée (joueur maximum AMI, c'est-à-dire «attaque»; joueur maximum ENNEMI, c'est-à-dire « défense », ou maximum de la somme des deux évaluations, c'est-à-dire « efficacité »). L'adversaire est un être humain qui n'a pas accès au listing de la machine. Les expériences ont montré que le programme succombait contre un joueur humain bien entraîné, mais après une défense honorable (de 15 à 20 coups). Par contre, un joueur humain débutant ou distrait est presque toujours battu.

Le programme a alors été réécrit pour la machine I.B.M.-7090, mais ici, pour épargner du temps machine, les deux adversaires sont des programmes. La tactique étant la suivante

Si le maximum de la fonction d'évaluation correspondant à l'attaque est supérieur à un certain seuil, on joue ce coup, sinon on joue le coup de défense. Si le coup de défense lui-même n'atteint pas ce seuil, on joue le coup d'efficacité. Il est alors facile d'opposer deux programmes identiques, sauf en ce qui concerne la valeur du seuil. Nous l'avons divisé en trois catégories : BAS, MOYEN, ÉLEVÉ et joué 9 parties qui ont donné les résultats suivants (fig. 25).

Ces résultats indiquent bien que la recherche en est à un stade transitoire : il existe des programmes qui jouent assez bien, mais qui ne gagnent pas à tout coup. Cependant, il est probable que la construction d'un tel programme «champion du monde » ne rencontrerait pas de difficultés insurmontables.

De même pour le jeu de dames (ou sa version anglo-saxonne dénommée *checkers) ;* des programmes ont été écrits qui donnent des performances excellentes.

Cet exemple est particulièrement intéressant, car en plus d'un programme de jeu proprement dit, comprenant un programme de lecture

1er joueur \ 2e joueur	B	M	E
B	nulle	nulle	B 73e coup
M	M 71e coup	M (2) 172e coup	M 128e coup
E	B 58e coup	M 82e coup	E (1) 103e coup

FIG. 25. — Quelques résultats expérimentaux relatifs à la simulation du jeu de Go-Bang

du damier analogue à notre LG ci-dessus et un programme d'évaluation de la position tenant compte du bilan des pions pris par les adversaires en présence, de l'avantage en position, etc., le programme a été muni de possibilités *d'apprentissage* (ce programme a été décrit dans plusieurs publications de A. Samuel, dont la plus importante est reproduite dans l'ouvrage édité par Feigenbaum et Feldman et cité dans notre bibliographie).

On se souvient qu'au début de ce chapitre nous avons écarté de notre enquête la simulation du phénomène psychologique de l'apprentissage. Mais l'apprentissage de l'automate simulant le joueur, tel qu'il est pratiqué dans les expériences de Samuel, n'a aucun rapport avec celui que les psychologues s'efforcent d'analyser chez l'homme. Il s'agit simplement de l'utilisation de la capacité de mémoire de l'automate pour le stockage des parties jouées et le décompte des

performances de l'automate. La fonction de préférence à laquelle nous avons fait allusion plus haut contient certains paramètres numériques qui peuvent être changés et pour lesquels on peut prévoir une procédure d'adaptation en fonction des résultats acquis.

Par contre, dans le domaine plus spectaculaire des échecs, malgré des efforts considérables, de nombreuses équipes (Shannon, Turing, Ulam, Bernstein, Newell-Shaw-Simon, Shura-Bura, Euwe), aucun résultat vraiment significatif n'a été atteint, du point de vue du jeu lui-même (1). Par contre, les difficultés rencontrées, les moyens mis en oeuvre pour les attaquer, sont une contribution fondamentale au développement de l'intelligence artificielle. Nous aurons l'occasion d'y revenir au chapitre VI (2).

(1) Tout récemment le soviétique Adelson-Vilsky a développé un programme efficace qui a battu, après une partie honnête, le programme américain dû à l'équipe de J. McCarthy (cf. l'article « Automates » dans le récent ouvrage de F. La LIONNAIS et E. MAGOT, *Dictionnaire des échecs,* Paris, Presses Universitaires de France, 1967, P. 25).

(2) Depuis que ces lignes ont été écrites nous avons abordé la simulation de jeux nouveaux sur machine IBM 360/50. Nous avons été amenés d'ailleurs *à inventer* des jeux spécialement adaptés à notre enquête. (Cf. notre article : The gaine of FIB an experiment in artificial intelligence, en collaboration avec H VAN HEDEL, soumis pour publication).

CHAPITRE IV

La raison

Dans le chapitre II, nous avons montré que la nécessité de codifier les informations émises par le monde extérieur, nécessité commune aux être organisés et aux automates, permettait de formuler les problèmes de l'intelligence dans un langage relativement simple et bien défini, et surtout protégé contre le danger de spéculations psychologiques ou plutôt pseudo-psychologiques.

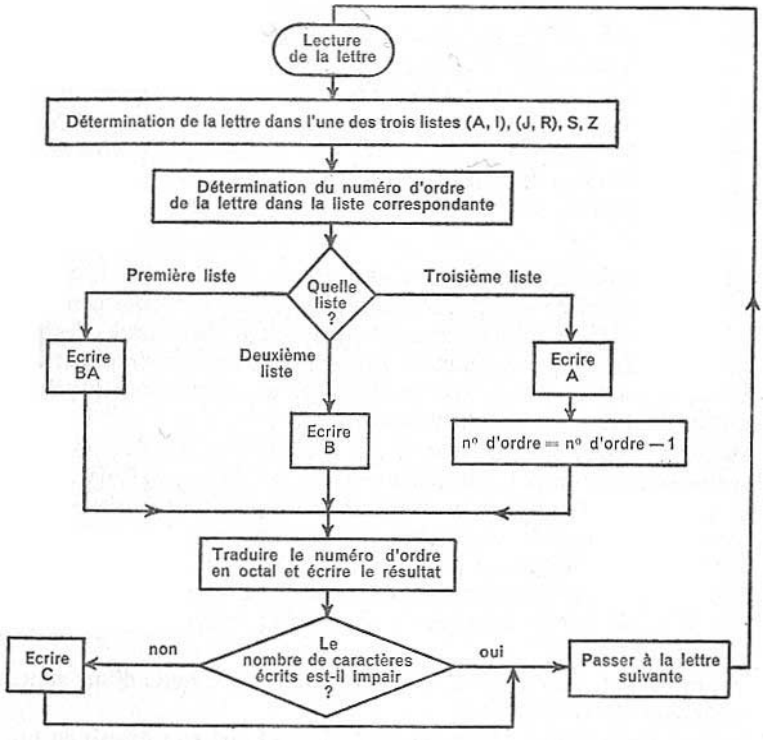
Au cours du chapitre III, nous avons illustré cette thèse en analysant quelques jeux, dont la simulation a été tentée sur calculatrice (et en particulier l'un d'entre eux, le jeu de Go-Bang). Nous avons observé alors que la simulation des jeux nécessitait tout d'abord une double codification : celle des règles du jeu, et celle des états successifs du jeu (position des pions sur un damier, par exemple) ; puis le choix d'une stratégie et sa mise en oeuvre sous forme de programme.

Dès ce niveau, il est donc clair qu'on a dépassé le stade d'une simple succession de codifications.

En fait, dès l'exemple du chapitre II, figure 7, on se trouvait en présence de deux attitudes possibles, pour construire une traduction automatique d'un code dans l'autre.

C'est ainsi que la transformation de l'alphabet romain dans le code octal BCD pourrait s'effectuer par le simple programme

Mais, il serait également possible, exploitant la structure particulière du code BCD, telle qu'elle apparaît clairement sur la figure 7, de construire le programme suivant



Ftc. 26. — Organigramme d'un programme hypothétique permettant de traduire le code alphabétique usuel dans le code BCD
A la fin de la 3e ligne, lire : (A, I), (J, R), (S, Z)

Un examen attentif de la figure 7 permet de comprendre en quoi la « structure du code a été effectivement utilisée : la correspondance entre le code alphabétique et le code BCD n'est en effet pas quelconque. Les codes BCD ayant même préfixe en A et B forment des

groupes qui correspondent à des lettres qui se suivent dans l'alphabet; par ailleurs, le caractère C n'apparaît que pour rendre impair le nombre total de caractères.

Quelle est la différence entre le programme esquissé dans la figure 26 et la simple consultation de table ? D'une part, dans le cas d'un alphabet très grand il faudrait une table très grande, donc qui utiliserait une grande place dans la mémoire. En plus, pour chercher l'équivalent d'un caractère donné, il faudrait, en moyenne, consulter la moitié de la table, donc utiliser l'automate pendant un temps qui pourrait devenir assez long.

par contre le programme lui-même est beaucoup plus petit que dans le cas de la figure 26.

Bien entendu notre exemple n'est pas significatif en lui-même car pour un alphabet de 26 lettres la consultation de table est évidemment la procédure la plus convenable. Mais il permet, pensons-nous de comprendre comment, dans des cas plus compliqués, il deviendrait possible de gagner du temps (et de la place dans le mémoire) en remplaçant l'exploration exhaustive d'une table par un algorithme qui exploite les particularités d'une structure.

Et puisque, au début de ce chapitre, nous indiquions que la simulation des jeux impliquait plus qu'une simple succession de codifications, nous pouvons souligner maintenant qu'à son tour la codification est souvent plus qu'une suite de substitutions : elle est un *arrangement structuré* de substitutions. Dans ces conditions, le jeu et sa simulation ne font qu'amplifier l'aspect structuré de ces manipulations de symbole.

Et c'est pourquoi les jeux semblent être l'apprentissage nécessaire de la raison. Si le mot même de «raison » évoque aussitôt ceux de « lois»., de « règles », etc., c'est bien parce que son fonctionnement n'est pas autre chose que l'actualisation, sur une donnée codifiée, d'un système ordonné et concerté de substitutions interdépendantes. La situation que nous avons rencontrée à l'occasion de l'étude des jeux n'est donc qu'une étape intermédiaire entre la codification élémentaire, du type substitution, et la manipulation complexe de symboles, où la manipulation elle-même compte plus que le symbole, comme c'est finalement le cas pour l'activité mathématique supérieure.

Sans négliger l'importance de l'aspect « sémantique » dans le traite-

ment de systèmes linguistiques (systèmes simples comme celui que nous avons associé à un jeu à deux personnes, ou systèmes complexes comme ceux que nous évoquerons dans le chapitre suivant à propos du langage naturel), le premier élément dans l'analyse de tels systèmes et la mise en évidence et l'exploitation de structures de type formel.
Et c'est pourquoi les efforts pour l'automatisation du labeur mathématique, tels qu'ils apparaissent sous la forme de programmes pour la démonstration automatique de théorèmes, en diverses branches de la logique mathématique ou de la mathématique plus traditionnelle, apparaissent, d'un certain point de vue, comme le point le plus avancé dans la tentative de développement d'une intelligence artificielle.

Aussi n'aborderons-nous ce terrain difficile que par petites étapes, en allant du simple au complexe. Et, ce faisant, nous suivrons naturellement le fil historique des automates « raisonneurs ».

BREF RETOUR AU MOYEN AGE

Il est bien connu que la logique du Moyen Age, la scolastique, était essentiellement un *jeu* de combinaisons de raisonnements standards appelées « syllogismes », qui étaient des suites de trois propositions (la 3e étant la conclusion, les deux premières les prémisses).
On a en mémoire l'exemple fameux

- proposition i : Tous les hommes sont mortels
- proposition 2 : Socrate est un homme

] prémisses

- proposition 3 : Socrate est mortel — conclusion

Le Moyen Age considérait déjà ces exercices déductifs comme la mise en oeuvre d'un *mécanisme* intellectuel (d'ailleurs, on désignait la logique d'Aristote sous le nom *d'Organon,* qui signifie « instrument»). Aussi avait-on déjà (contrairement à Aristote) développé un symbolisme à usage essentiellement mnémotechnique.
On désignait

- par la lettre A les propositions « universelles affirmatives », telles que : « Tous les X sont des Y » ;
- par la lettre E les propositions « universelles négatives », telles que : « Aucun des X n'est parmi les Y»;

– par la lettre I les propositions « particulières affirmatives », telles que : « Quelques-uns des X sont des Y » ;
- par la lettre O les propositions « particulières négatives », telles que : « Quelques-uns des X ne sont pas des Y ».

Les syllogismes étaient des suites de trois propositions du type précédent : en remplaçant les propositions par le symbole A, E, I ou O de leur *mode,* on obtient des expressions du type AAA, EIO, etc.
Cependant, il convient encore de distinguer la *forme* du syllogisme qui est définie par la correspondance des éléments des propositions constituant le syllogisme.
C'est ainsi qu'un syllogisme du type

tous les Y sont des X
tous les Z sont des Y
tous les Z sont des X

est de la première forme. On a les définitions suivantes

1re forme : suite YX, ZY, ZX
2e ___ : ___ XY, ZY, ZX
3e ___ : ___ YX, YZ, ZX
4e ___ : ___ XY, YZ, ZX

Pour chacune de ces formes, les diverses combinaisons de trois lettres parmi A, **E, I,** O, ne sont pas acceptables. Mais on disposait d'un texte pseudo-latin qui permettait de les retrouver; il s'agit du texte fameux

« bArbArA, cElArEnt, dArII, fErlOque prioris.
cEsArE, cAmEstrEs, fEstlnO, bArOkO, secundae », etc.

indiquant que les syllogismes AAA, EAE, AII, EIO, étaient possibles dans la première forme, EAA, AEE, EIO, AOO dans la seconde, etc.
Mais ce texte ne faisait que faciliter la mise en mémoire de la liste *exhaustive* des syllogismes acceptables. A l'énoncé des prémisses d'un syllogisme particulier, il fallait balayer l'ensemble des formes ainsi mémorisées pour trouver celle qui s'appliquait et en déduire la conclusion.
Aussi le mécanisme (purement graphique) imaginé par Raymond Lulle dès 13oo, et auquel nous ferons allusion au cours du chapitre VIII, n'était pas une machine à penser, mais seulement un

mode de représentation de concepts et de catégories dont Leibniz devait faire une critique serrée. Bien entendu, Leibniz, dans son effort pour construire une « caractéristique universelle », c'est-à-dire une langue artificielle dont la grammaire complètement explicitée et purement logique permettrait la découverte et l'énoncé automatique des propositions vraies, se servit-il des idées de Lulle et de ses successeurs comme d'un tremplin pour des tentatives nouvelles représentation graphique des figures du syllogisme, utilisation d'opérations du type arithmétique pour exprimer les combinaisons logiques. Pourtant, ses efforts n'aboutirent pas, faute d'utiliser la formalisation efficace et il est intéressant de noter que, malgré son désir passionné de mettre en pratique ses idées relatives à la combinatoire et à la caractéristique, Leibniz dessina les plans d'une machine arithmétique, mais pas d'une machine logique.

Cent cinquante ans plus tard, G. Boole, reprenant l'étude des syllogismes de l'école, mais dans un esprit plus systématiquement algébriste, trouvait la clé qui allait permettre la mécanisation.

Utilisant les notions et représentations ensemblistes introduites dans

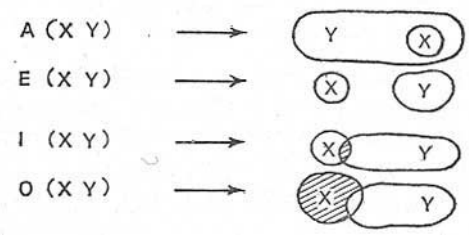


FIG. 27. - Mise en correspondance des prédicats traditionnels du syllogisme avec une représentation ensembliste
On remarquera que les parties hachurées sont nécessairement non vides

le chapitre II, on voit que les modes du syllogisme reçoivent l'illustration élémentaire reproduite dans la figure 27.

Désignons par x l'ensemble des êtres qui sont des x, par 1 - x l'ensemble des êtres qui ne sont pas des x, et par o l'ensemble vide ; utilisons les symboles d'addition et de multiplication pour exprimer

la réunion et l'intersection des ensembles. On obtient la correspondance

$$A(X,Y) \rightarrow x.y = x \qquad (\text{ou } x.(1\text{-}y) = 0)$$
$$E(X,Y) \rightarrow x.y = 0$$
$$I(X,Y) \rightarrow x.y \neq 0$$
$$O(X,Y) \rightarrow x.(1\text{-}y) \neq 0$$

Il est donc possible de procéder à une *réduction* des syllogismes en expressions algébriques, puis en équations susceptibles d'être résolues par des moyens canoniques. Chemin faisant, on s'aperçoit que les formes traditionnelles du syllogisme représentaient des combinaisons assez particulières dont le respect trop strict fut la cause de l'échec de Leibniz.

Par contre, dès que l'algébrisation est accomplie, la mécanisation est possible. Elle est effectivement réalisée par Jevons dès 187o au moyen d'un système de plaquettes et de leviers. On pourrait, de ce point de vue, considérer la machine de Jevons comme une première réalisation pratique de l'intelligence artificielle. En réalité, il ne s'agit que d'une machine à calculer élémentaire comme celles de Pascal et de Leibniz, avec cette différence qu'elle effectue des opérations logiques à la place des opérations arithmétiques usuelles.

Nous avons cependant tenu à développer un peu cette phase préliminaire pour mettre en valeur, une fois de plus, l'importance du choix d'une formalisation heureuse. Ce choix est au moins aussi important pour la mise en machine des problèmes abstraits que pour leur prise en charge par nos cerveaux humains.

MISE A JOUR DES AUTOMATISMES DANS LES PROCESSUS DE LA DÉMONSTRATION

Dans le cas des jeux à information complète (c'est-à-dire pour lesquels le hasard n'intervient pas), le joueur se trouve en présence d'une situation donnée et il dispose d'un ensemble de règles. Son initiative est donc limitée de deux façons et son talent consiste à effectuer le bon choix parmi les possibilités qui lui sont laissées.

Il en va au fond de même du mathématicien : placé en face d'un ensemble d'axiomes et de théorèmes et disposant d'un certain nombre

de procédés déductifs, il a le choix entre différentes voies selon qu'il applique telle ou telle règle à tel ou tel sous-ensemble d'axiomes ou de théorèmes. Ici la victoire, c'est évidemment la découverte et la démonstration d'un théorème.

S'il est possible de simuler le comportement du joueur, on peut s'imaginer sans peine qu'il devrait être également possible de simuler le comportement du mathématicien, ou, plus précisément, de programmer un automate qui produira des théorèmes (sans qu'il y ait de réelle similitude entre son fonctionnement interne et celui du mathématicien). Ici encore, une seule condition préalable : la formalisation.

On sait que l'usage des symboles est relativement tardif : les textes antiques n'utilisent que des mots du langage usuel pour désigner les êtres mathématiques tels que angles, polygones, etc. C'est seulement à partir de la Renaissance que l'usage des symboles se développe pour désigner les êtres, les opérations sur ces êtres, etc.

Mais, en fait, les textes mathématiques demeurèrent essentiellement écrits en langage naturel.

C'est seulement à la fin du xixe siècle et dans les premières années du xxe que certaines difficultés rencontrées dans des branches nouvelles de la mathématique, et en particulier dans la théorie des ensembles, jetèrent des doutes sur la validité de l'emploi du langage naturel et amenèrent la recherche d'une formalisation complète des étapes les plus petites du raisonnement.

Le but de ces recherches était la formalisation de déductibilité. Ceci fut obtenu de la façon suivante : une fois donnée la notion d'alphabet, de formules et de règles de déduction, ainsi qu'on l'a vu au chapitre II, on distingue une famille A de formules qui sont les *axiomes* du système formel. Soit alors K un autre ensemble de formules. On dit qu'une formule f est *déductible* à partir de A au sein du système formel considéré, si elle résulte d'un nombre fini d'applications des règles de déduction à des formules de $K \cup A$, ce que l'on écrira

$$K \vdash f$$

Le théorème fondamental de la déduction dû à Herbrand (1930) s'énonce ainsi

Si K est composé des formules f, g, etc., n, on a

$$K \vdash s \text{ si et seulement si } (f \,\&\, g \,\&\, \ldots \,\&\, n) \rightarrow s.$$

Le problème de la décision est alors le suivant : étant donné une famille K et une formule s, trouver si on a ou non $K \vdash s$, et, dans l'affirmative, présenter la démonstration.

Le résultat fondamental, en ce qui concerne le problème théorique de la décision, est le suivant, dû à Church (1936)

Il n'existe pas de procédé automatique permettant de résoudre le problème de la décision, dans le cas où le système formel est celui que nous avons décrit au chapitre II sous le nom de « calcul des prédicats ».

Nous voyons ainsi apparaître une première différence entre le problème de la démonstration automatique des théorèmes et celui de la simulation des jeux. Nous avions souligné, au cours du chapitre précédent, que le fait de disposer d'un théorème d'existence pour une stratégie gagnante n'impliquait pas la découverte automatique d'une telle stratégie. Ici nous pouvons préciser davantage : pour le « jeu » de la démonstration automatique une telle «stratégie gagnante» *ne peut pas* être construite automatique.

En d'autres termes, il n'existe pas d'automate A muni d'un programme P tel que, si l'on introduit une formule quelconque φ comme donnée, l'automate s'arrêtera au bout d'un temps fini et fournira l'un des résultats suivants : « φ est un théorème », « φ n'est pas un théorème ».

Bien entendu, de même que, dans le cas des jeux, l'absence d'une stratégie gagnante toute prête n'a pas empêché les chercheurs de construire des programmes qui gagnent « le plus souvent possible », de même ce théorème (ou plutôt ce « métathéorème », puisqu'il s'agit d'un théorème sur les théorèmes) n'a pas empêché que l'on construise des programmes qui démontrent des théorèmes « le plus souvent possible ». Après tout, nous savons fort bien que les hommes ne gagnent pas toujours, qu'ils ne réussissent pas toujours à démontrer des théorèmes (même si l'énoncé est apparemment simple comme c'est le cas pour le célèbre théorème de Fermat).

Pour mieux apprécier les difficultés qui surgissent ici, il n'est pas mauvais de faire une brève incursion dans le domaine de la logique mathématique qui n'est pas seulement la source des résultats métathéoriques négatifs, mais aussi des techniques qui permettent de contourner, dans une certaine mesure, ces résultats.

Pour cela nous remarquons que si les mathématiques sont des systèmes formels, ainsi que nous l'avons souligné au cours du chapitre II, elles se sont constituées historiquement en vue de correspondre à une certaine intuition des mathématiciens, intuition qui se référait à certains objets indépendants du système formel, même s'il s'agissait d'objets très différents de ceux qui peuplent notre vie quotidienne.

C'est ainsi que la théorie des ensembles, à laquelle nous faisions allusion au cours du chapitre II, n'est pas seulement un langage dont l'alphabet comprend des lettres diverses, des signes tels "$\cup$", "$\cap$", "$\subset$", "$\in$" , etc. C'est aussi une théorie qui s'efforce de décrire les propriétés de familles d'objets dont les propriétés particulières sont toutes passées sous silence, sauf celle qui consiste pour eux à exister.

Il est donc naturel de mettre en correspondance le point de vue qui était essentiellement le nôtre au cours du chapitre II et qui consistait à définir les conditions de construction de formules « bien faites » à partir d'un alphabet - c'est-à-dire le point de vue *syntactique* –, avec un point de vue dans lequel ces formules expriment les propriétés d'un univers extérieur - c'est ce qu'on appelle le point de vue *sémantique.*

Plus précisément le point de vue sémantique se développe en associant au système formel une *interprétation* et un système de *valuations.*

L'interprétation consiste à associer au système un ensemble E et l'ensemble B constitué des deux éléments { vrai } et { faux } puis à faire correspondre aux variables et constantes du système formel des éléments de E, aux prédicats du système des fonctions des éléments de E prenant leurs valeurs dans B, etc.

La valuation s'obtient en décomposant les formules en composantes élémentaires, utilisant l'interprétation des prédicats puis appliquant, pour la composition des composants élémentaires, les règles traditionnelles du calcul des prédicats. On obtient ainsi une application de l'ensemble des formules du système dans l'ensemble B. Un ensemble E est un *modèle* de la formule φ si la valuation obtenue ainsi pour φ est telle que

$$w(\varphi) = \{\text{vrai}\}.$$

Si K et L sont deux ensembles de formules on note par :

$$K \models L$$

le fait que tout modèle de toutes les formules de K est modèle d'au moins une formule de L.

Le théorème de Gödel-Herbrand établit alors la relation entre la version syntactique et la version sémantique en montrant que les deux propositions suivantes sont équivalentes

a) $\vdash \varphi$ (φ est déductible);
b) $\sim \varphi \models$ (la négation de φ est irréalisable).

Pour démontrer ce théorème (dont nous avons d'ailleurs donné une version incomplète, pour simplifier), il faut construire un ensemble E destiné à servir de base au modèle et c'est la construction de cet ensemble qui constitue d'une part la partie essentielle de la démonstration et d'autre part le point de départ des commentaires que nous aurons à proposer par la suite (en particulier dans le chapitre VI).

Cet ensemble est ce que l'on appelle *l'univers de Herbrand* associé à la formule φ. Il est obtenu par la réunion d'une suite infinie d'ensembles que l'on dénote par H_0, H_1, H_2,

L'ensemble H_o, (que l'on appelle premier niveau de l'univers de Herbrand) comprend les constantes figurant dans la formule cp. H_1 est formé à l'aide de certains couples de constantes (qu'on appelle fonctions de Skolem), H_2 au moyen des fonctions de Skolem de $H_o \cup H_1$, etc. Il est facile de comprendre que le nombre des éléments qui composent les ensembles H,, croît extrêmement vite avec n, même si H» contient très peu d'éléments.

Sans entrer dans des détails trop techniques, nous indiquerons que les procédures de décision (ou de semi-décision) consistent à examiner les éléments des univers de Herbrand Ho, H1, etc., et à vérifier qu'ils satisfont certaines conditions.

Dans le cas général il faudrait examiner H,, et faire tendre n vers l'infini ce qui évidemment est irréalisable. Cependant, si l'on ne raisonne pas sur une formule cp quelconque, mais qu'au contraire on en explicite - au moins partiellement - la forme, il n'est pas impossible de voir l'examen des H,, se terminer relativement vite et dès lors la démonstration automatique devient possible.

Soit par exemple la formule (logique)

$$(\exists x)\ (\forall y)\ P(x, y) - (\forall v)\ (\exists u)\ P(u, v)$$

qui (conformément aux définitions rappelées dans le chapitre II) peut se lire : la proposition : «il existe un objet x tel que pour tout la propriété P (x, y) soit satisfaite » entraîne la proposition : « pour tout v il existe un u tel que la propriété P (u, v) soit satisfaite».
Cette formule φ est donc partiellement explicite (partiellement seulement car la propriété P peut être quelconque). Mais en tout cas l'examen des éléments de l'univers de Herbrand H_2 associé à φ suffit ici à achever la démonstration (1).

AUTOMATISATION D'UNE DEMONSTRATION ARITHMETIQUE

Le lecteur trouvera peut-être que les développements que nous venons de présenter sont apparemment fort éloignés du raisonnement mathématique tel qu'on le rencontre habituellement dans les traités. Dans les livres, les cours, dans les exposés que font les mathématiciens, les discours se présentent sous la forme d'un mélange intime de symbolisme et de langage usuel.
En explicitant, comme nous l'avons fait, les étapes logiques du raisonnement, dans leur détail le plus minime, et en les formalisant, nous ne nous sommes écartés du langage usuel du mathématicien que pour mieux nous rapprocher de celui de l'automate et pour cela la formalisation logique est un intermédiaire excellent (2).
Toutefois, à l'intention des lecteurs que la logique mathématique dérouterait un peu, nous avons tenu à développer deux exemples de raisonnement mathématique proprement dit, suffisamment simples pour que l'armature logique puisse être mise entre parenthèses.

(1) Ce paragraphe s'est longuement inspiré de l'enseignement professé par D. BERG à l'Université libre de Bruxelles.

(2) Cf. l'argumentation de Hao WANG dans son article Mechanical mathematics and inferential analysis dans l'ouvrage de P. BRAFFORT et D. HIRSCHBERG cité en bibliographie.

Nous allons chercher tout d'abord à démontrer la formule arithmétique très simple

$$2m + 2n = 2(m+n)$$

qui est un cas particulier de la distributivité de la multiplication par rapport à l'addition, en supposant que seule l'addition est donnée et que l'on dispose des transformations (présentées sous forme d'opérateurs)

A : $2x \to x+x$	G : $(x+y) + z \to x + (y+z)$
S : $x+x \to 2x$	D : $x + (y+z) \to (x+y) + z$
C : $x+y \to y+x$	R : $x+y \to z$

Si nous appliquons chacune de ces transformations à la formule initiale, on a (en convenant de n'appliquer chaque transformation qu'une fois, lorsqu'on parcourt la formule de gauche à droite)

$$A[2m+2n] = (m+m)+2n \quad C[2m+2n] = 2n+2m$$
$$A[A[2m+2n]] = (m+m)+(n+n)$$
$$G[A\ [A\ (2m+2n)]] = m + (m + (n + n)),\ \text{etc.}$$

Finalement, on vérifiera que

$$S.R.G.C.G.D.A.A.\ [(2\ n)] = 2\ (m + n).$$

On a donc bien trouvé une démonstration de la propriété

$$2m+2n=2(m+n).$$

Cependant, la recherche *automatique* de cette démonstration est loin d'être immédiate, car la machine n'a pas de raison particulière pour essayer telle transformation plutôt que telle autre. La seule possibilité est donc d'essayer toutes les transformations de la liste et d'éliminer les chemins circulaires au fur et à mesure qu'on les détecte. Tôt ou tard, on rencontre la suite de transformations qui donne le résultat et le programme s'arrêtera. Bien entendu, il faut garder en mémoire les essais infructueux afin de ne pas s'y engager à nouveau.
Pour cela, il suffit d'associer à toute expression obtenue une liste de transformations qui lui sont applicables, et d'affecter à chaque transformation une étiquette portant 0 ou 1, suivant qu'elle a été

déjà essayée ou non au cours de l'exécution du programme. Lorsqu'une expression déjà rencontrée est obtenue, on rebrousse chemin vers l'expression précédente et on utilise la première transformation possible qui n'a pas encore été essayée.

Le programme prend alors l'allure de la figure 28.

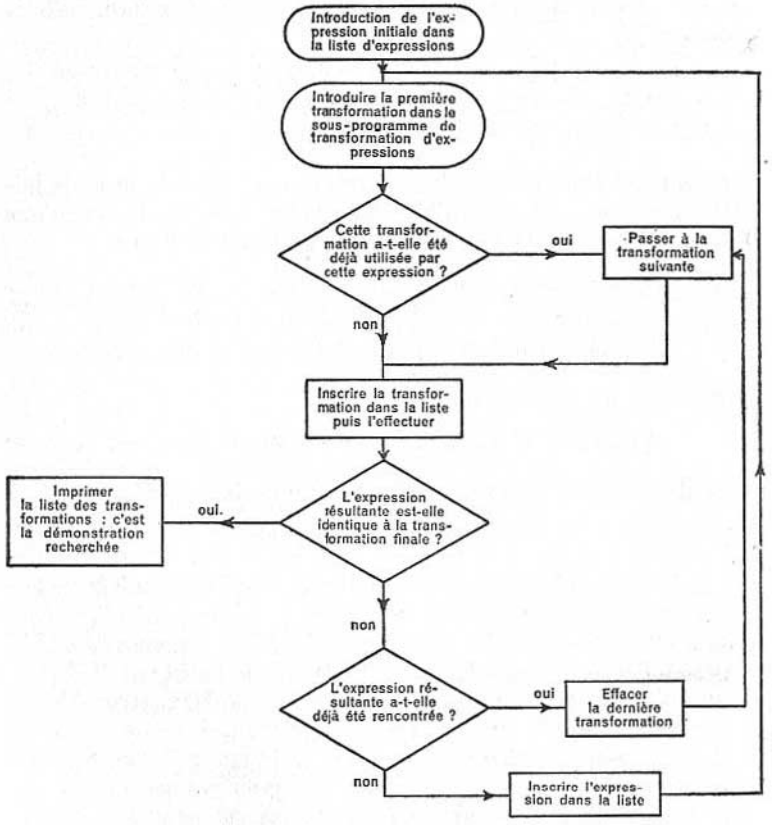


FIG. 28. - Organigramme correspondant à la recherche d'une démonstration automatique de la formule $2^m + 2^n = 2^{(m+n)}$

En somme, la tâche du mathématicien dans ce problème, c'est de trouver un mot composé des lettres A, S, C, G, D, et qui exprime le contenu de la démonstration.

Plusieurs mots (et même une infinité d'entre eux) sont possibles. Le plus court a 7 lettres. Mais il n'est pas évident que le programme va s'arrêter sur cette combinaison ou sur une autre de même longueur. Il est même évident que la suite des essais effectués dépendra de l'ordre adopté pour les transformations.

Il se pourrait même que le programme ne s'arrête jamais, construisant sans cesse des expressions nouvelles sans se rapprocher de l'expression finale recherchée. Dans l'exemple que nous avons choisi, cette dérive formelle n'a pas lieu parce que le nombre d'expressions que l'on peut construire est limité d'avance par la nature même des transformations autorisées.

On voit donc, sur cet exemple pourtant bien simple, la complexité des problèmes de la démonstration automatique des théorèmes, même dans le cas où un arsenal logique important n'a pas besoin d'être mis en oeuvre. Nous aurons l'occasion d'y revenir au chapitre sur la complexité. Nous verrons alors s'en dégager des problèmes et des propriétés de nature tout à fait générale.

AUTOMATISATION D'UNE DEMONSTRATION GEOMETRIQUE

Qu'en arithmétique et en algèbre les démonstrations soient des jeux de substitutions dans les formules n'a rien de surprenant. Il peut donc être intéressant d'examiner ce qui se passe dans le domaine de la géométrie élémentaire.

Ici encore, nous allons présenter un exemple assez détaillé : celui des constructions géométriques que l'on peut effectuer avec la règle et le compas. Dire qu'une série d'opérations permet d'effectuer une construction donnée, c'est en effet énoncer un théorème. Découvrir et justifier une construction, c'est donc découvrir une démonstration. Mais ici, ces théorèmes sont en quelque sorte des *programmes.*

Notre exemple est donc à la fois un exemple de programme de démonstration automatique et un programme de construction de programmes.

Nous allons tout d'abord définir le langage de programmation correspondant aux opérations de construction géométrique (en nous limitant d'ailleurs aux constructions par la règle seulement). Notre langage sera composé d'instructions de deux types

a) Instructions d'initialisation, servant à définir des éléments de la figure initiale et dont le format sera

n	D			symbole

et dont l'interprétation est : « l'instruction *n* consiste à définir une droite appelée symbole ».

Dans la même catégorie, on aura les instructions

n	P			symbole

qui s'intitule : « la *ne* instruction consiste dans la définition d'un point appelé symbole » ;

n	PD	*m*		symbole

qui s'interprète : « la *ne* instruction consiste à choisir un point sur la droite définie dans l'instruction m et qu'on appellera symbole ».

b) Instructions de construction proprement dite dont les formats seront

n	R	*p*	*q*	symbole

qui s'interprète : « la *ne* instruction consiste à tracer une droite joignant les points définis dans les instructions *p* et *q* et qu'on appellera symbole.

n	X	*p*,	*q*	symbole

qui s'interprète : « la *ne* instruction consiste à définir le point situé à l'intersection des droites définies dans les instructions *p* et *q* et qu'on appellera symbole ».

Considérons, pour fixer les idées, la construction, sur une droite 1 où l'on s'est donné 3 points A, B et C, d'un point D qui, conjugué à C, divise harmoniquement le couple A, B (c'est-à-dire tel que

$$\frac{AC}{CB} / \frac{AD}{DB} = -1$$

Le lecteur vérifiera (en s'aidant de la figure 29) que la construction est bien donnée par le programme suivant

Initialisation

1	D		1
2	PD	I	A
3	PD	I	B
4	PD	I	C

Construction

5	P		E
6	R	2,5	AE
7	R	3,5	BE
8	R	4,5	CE
9	PD	6	F
10	R	3,9	BF
11	X	8,10	G
12	R	2,11	AG
13	X	7,12	H
14	R	9,13	FH
15	X	1,14	D

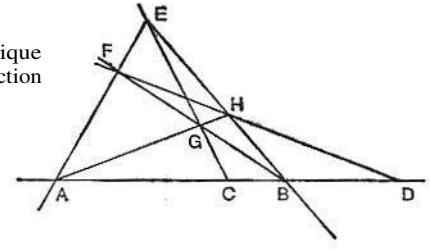


FIG. 29. - Figure géométrique correspondant à la construction

Bien entendu, la construction n'est fondée, et ne devient donc une démonstration, que si l'on s'appuie explicitement sur un théorème de géométrie

« La paire des points de la diagonale d'un quadrilatère complet qui ne sont que sur un des côtés du quadrilatère divise harmoniquement la paire des points de cette diagonale qui sont aussi à l'intersection de deux côtés du quadrilatère. »

Il suffit alors de remarquer que AB est une diagonale du quadrilatère complet EFGH.

Ces remarques préliminaires nous permettent d'imaginer comment un programme de recherche automatique de constructions de figures géométriques au moyen d'une règle pourrait être construit.

Le schéma général sera le suivant

CR Programme de recherche de constructions géométriques utilisant la règle	algorithme : élaboration d'une suite d'instructions du type D, P, PD, R et X ; documentation de base : théorèmes de géométrie élémentaire ; données : énoncé de la figure initiale et des propriétés requises pour la figure finale ; résultat : énoncé d'une liste d'opérations de construction et des théorèmes qui en assurent le succès.

On voit bien ce que notre problème a de commun avec le précédent ici, les figures remplacent les expressions et les opérations de construction remplacent les transformations algébriques. Mais cette dernière correspondance n'est que partielle : dans le cas algébrique, les transformations possibles exprimaient les théorèmes que l'on mettait en application. Par contre, dans le cas géométrique, les théorèmes sont formulés dans un *langage* qui n'est pas identique à celui des constructions (bien qu'ils aient quelques parties communes).

Il est donc important de dégager ce langage et de le formaliser. On met alors en évidence

a) des concepts

1) point	P
2) droite	D
3) triangle	T
4) quadrilatère, etc.	Q

b) des relations

- unaires, telles que :
  - à l'infini (valable pour les concepts 1 ou 2) ;
  - isocèle (valable pour 3) : Is (T), etc. ;
- binaires, telles que
  - situé sur (valable pour un couple de concepts du type 1,2) S (P, D) ;
  - passant par (valable pour un couple de concepts du type 2, 1) : A (D, P) ;
- ternaires, telles que

  à l'intersection de (valable pour un couple de concepts du type 1, **2,** 2) : X (P, D, D) ;
  - reliant (valable pour un couple de concepts du type 2, **I,** I) R (D, P, P), etc. ;

c) des fonctions : numériques, comme la longueur d'un segment

$$L\ (P1, P2).$$

Des définitions plus complexes et les théorèmes portant sur les concepts et relations primitives, ainsi que sur les êtres construits par définition, s'exprimeront à l'aide des symboles de la logique élémentaire que nous avons introduits dès le chapitre II.

C'est ainsi que la notion de diagonale d'un quadrilatère complet pourra s'obtenir par

$$\begin{aligned}\text{Diag } (D, Q) \_ \ & (D_1 \text{ e } Q) \,\&\, (D_2 \text{ E } Q) \,\&\, (D_3 \text{ E } Q) \,\&\, (\mathbf{D_4} \text{ E } Q) \,\&\\ & X\ (P_1, D_1, D_2\} \,\&\, X\ (P_2, D_3, D_4) \,\&\\ & R\ (D, P_i, PD.\end{aligned}$$

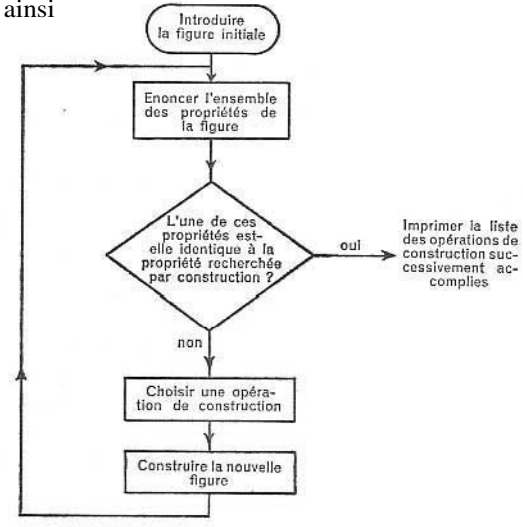
La conjugaison harmonique de deux couples de points (P_1, P_2), (P_3, P_4) s'écrit .

$$\begin{aligned}&\text{Conj Harm } (P_1, P_2 ; P_3, P_4)\\ &\quad S(P_1, D)\&S(P_2, D)\&S(P_3, D)\&S(P_4, D)\\ &\quad \& \{ [D\ (P_1, P_3) : L\ (P_3, P_2)] : [L\ (P_1, P_4) : L \quad (P_4, P_3)] \ \} = -\ I.\end{aligned}$$

Le théorème cité se formule alors comme suit

$$[\text{Diag}(D, Q)] \,\&\, (D_1 \in Q) \,\&\, (D_2 \in Q) \,\&\, (D_3 \in Q) \,\&\, (D_4 \in Q)$$
$$\&\, X(Pl, Dl, D2) \,\&\, X(P2, D3, D4)$$
$$\&\, R(D, P_1, P_2)$$
$$\&\, X(P_3, D, R(X(D_1, D_3), X(D_2, D_4))$$
$$\&\, X(P_4, D, R(X(Dl, D4), X(D2, D3)))$$
$$\Rightarrow \text{Conj Harm}(P_1, P_2, P_3, P_4).$$

Le problème de la recherche automatique des constructions se précise alors ainsi



Ici, aux difficultés rencontrées précédemment, et qui concernent la possibilité de cyclage indéfini d'opérations, s'en ajoute une autre, celle contenue dans le rectangle « énoncer l'ensemble des propriétés de la figure ».

Ce sont les difficultés que nous avons évoquées plus haut, à l'occasion de notre discussion sur les programmes de logique automatique.

Pourtant, le fait d'avoir choisi un domaine assez restreint de la géométrie élémentaire nous permet d'entrevoir une possibilité de solution. Nous y reviendrons dans les chapitres suivants.

CHAPITRE V

Le langage

Au cours des deux chapitres précédents, nous avons indiqué que l'étude de systèmes tels que les jeux ou les disciplines mathématiques pouvait être considérée comme une première approximation de l'étude du langage, dans la mesure où ils forment eux-mêmes des langages relativement simples et en tout cas complètement formalisables. Nous avons cependant noté que, en particulier dans le cas des mathématiques, la formalisation complète est rarement effectuée et pose même parfois des problèmes insolubles.

Cela n'est pas étonnant puisque, revenant pour un instant au point de vue génétique, il est certain que les aptitudes au jeu, au raisonnement, etc., se développent de pair avec l'aptitude au langage et forment avec elle un ensemble difficilement séparable en parties.

A tous points de vue, le langage est ainsi le support essentiel du développement de l'intelligence et même de son existence. Il est donc obligé que le « front linguistique » de l'intelligence artificielle joue un rôle central dans notre propos. Succès et échecs de la linguistique automatique sont typiques du pouvoir et des limitations des procédés dont dispose à ce jour la discipline dont nous nous occupons ici.

Pour simplifier les choses, nous ne considérerons que les problèmes relatifs au traitement automatique de la langue écrite et l'exemple qui soutiendra nos diverses analyses sera emprunté à la seule langue française ; pourtant, on pourra se convaincre, chemin faisant, que

ces restrictions n'affectent pas sérieusement la validité des conclusions que nous pourrons tirer.

Le fait de nous restreindre au langage écrit simplifie considérablement le problème de *l'introduction des données.* Ce qui correspondra en effet, dans les programmes linguistiques, aux « données » des programmes décrits précédemment, ce sont les textes écrits (et transformés, pour la commodité de l'automate en cartes perforées, bandes perforées, bandes magnétiques, etc., grâce à l'un des codages décrits au début du chapitre II). Ce qui correspondra à ce que nous avons appelé plus haut la « documentation de base » ce sera l'ensemble des règles de grammaire, règles sémantiques, etc.

Toute la différence avec les problèmes traités précédemment réside dans l'absence jusqu'à ce jour d'une formulation exhaustive et même seulement cohérente, de ces différents systèmes de règles.

Il saute aux yeux qu'un texte écrit présente les propriétés qui nous ont servi à définir un monoïde au chapitre II. Prenons garde cependant que cette structure de monoïde existe simultanément à plusieurs niveaux avec les alphabets suivants (on pouvait en envisager davantage en découpant plus finement)

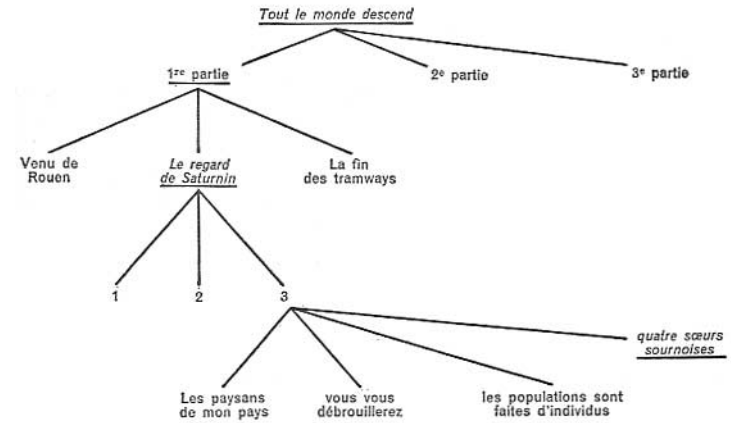
- un alphabet ordinaire ; les mots de ce monoïde sont les mots ordinaires ;
- un vocabulaire de mots ; les mots de ce monoïde sont les phrases du texte ;
- une liste de phrases ; les mots de ce monoïde sont les paragraphes ;
- une collection de paragraphes ; les mots de ce monoïde sont les chapitres.

Un même groupement de mots va donc pouvoir être considéré comme un élément d'un alphabet, ou un mot, ou une suite, suivant les niveaux auxquels on se place. On touche donc du doigt dès à présent - et sur un problème relativement élémentaire - la complexité du problème linguistique.

Cette complexité, ainsi que les moyens pour la réduire, apparaîtront mieux sur des exemples. Il nous a semblé intéressant de tirer nos exemples d'un texte unique, afin de faire mieux voir l'interprétation des problèmes et des méthodes.

UN TEXTE EXEMPLAIRE

Ce texte exemplaire sera un fragment de l'autobiographie poétique de Jean Queval : *Tout le monde descend.* Ce fragment se situe ainsi dans l'ouvrage



Il s'agit donc du dernier paragraphe de la troisième section du deuxième chapitre de la première partie de l'ouvrage ; nous le reproduisons ci-dessous *in extenso* (en *y* ajoutant une numérotation des phrases)

« 1. Quatre sueurs sournoises sur qui les garçons se retourneraient, s'ils l'osaient seulement, imitent la houle de la mer avec des effets de hanche. 2. Leur frère pêche à Terre-Neuve. 3. L'une d'elles a été vue avec trois garçons, à mi-falaise, entre des haies, tenant en rond quelque concile, assise à même les grosses et petites pierres dans une allée presque perdue du souvenir, tenant un concile en rond avec trois garçons, un soir de l'été. 4. A ces filles, parle ou bien ne parle pas, pour des raisons ignorées, la petite bonne de l'hôtel de la plage qui, deux fois la semaine, grossit au télescope le navire qui glisse sur la ligne d'horizon. 5. A bord, son fiancé est peut-être. 6. A tous ces êtres, Saturnin est le baromètre de l'humeur commune, comme un

jardinier de Giraudoux. 7. Cet ancien marin bricole dans les bistrots, pour une croûte et quelques verres. 8. Un mot de travers de l'un ou de l'autre, et on le chasse au balai, Saturnin. 9. Alors, au lieu d'aller dormir chez sa sueur, il découche pendant des jours. 10. C'est pendant une de ces périodes qu'eut lieu le grave concile d'une fille et de trois garçons qui probablement portait sur l'étrangeté des voies de la nature. ii. A ces moments-là, je le dis pour n'y plus revenir, tout est possible, vu le dérangement des astres. 12. Tout aussi bien, Saturnin revenu, l'ordre social se rétablit, comme par le retour de mouvement d'une grande horloge inconnue. 13. Ce n'est peut-être pas un fait de nature à simplifier la métaphysique, mais il en faut tenir compte, ou plutôt s'y soumettre, comme au temps même. 14. Le garde-champêtre le sait, Pierrot le facteur le sait, comme le savent la mère de Monsieur le Curé, et Monsieur le Curé lui-même, qui n'est pas parvenu à faire entrer cette donnée locale dans la véritable orthodoxie, au cours des promenades qu'il accomplit au bord de la mer, tout seul soir après soir. 15. La dernière fois que j'ai vu Saturnin, il pleuvait, et un couple s'était réfugié sous une porte - à bien regarder, on eût reconnu la championne de tennis et son admirateur ignoré -, et ce navigateur de la cloche, Saturnin, heureux seulement d'être là, allait son chemin, sous la pluie. 16. Voyant ce couple qui ne voulait pas être mouillé, il lui fit un discours de biais, c'est-à-dire à la troisième personne, celle qui concerne les tiers, et par là permet la généralisation. 17. « Z'ont point d'chance les estivants, c't'ânée », dit-il. i8. Puis ce furent les grandes tempêtes de l'automne. i9. Les dernières gens des vacances repartirent vers l'intérieur. 20. Les amoureux de six heures et demi du matin s'aiment maintenant dans le demi-jour gris venu de la mer, et le village se referme comme une huître des profondeurs, Saturnin régnant. »

Ce texte représente le paragraphe entier. Il comprend (si l'on limite les phrases aux textes compris entre deux points), 20 phrases. Les phrases contiennent environ 484 mots (« environ » fait allusion à différents points de vue possibles concernant les mots tels que : « mi-falaise », «c'est-à-dire», etc.). A ces mots s'ajoutent 48 virgules, i paire de tirets et une paire de guillemets.

Il s'agit donc d'un texte relativement court, et de composition tout à fait raisonnable. Et pourtant ce texte comporte, pour l'analyste,

un très grand nombre de difficultés - et c'est à ce titre que nous l'avons sélectionné. Nous sommes en effet bien loin du compte rendu, du procès-verbal objectif, voire du récit d'événements.

Tant que l'on demeurera au niveau du lexique ou même de la syntaxe, on ne rencontrera pas de difficultés plus grandes que pour l'analyse de tout autre texte, mais dès qu'on abordera le niveau sémantique et qu'on voudra pousser l'analyse en profondeur, on rencontrera des difficultés considérables.

Notre but n'étant pas seulement de décrire la situation actuelle dans ce qu'elle a de satisfaisant, mais aussi de souligner l'existence et l'importance des difficultés, nous développerons plus particulièrement ces derniers aspects, en utilisant toujours la représentation commode que nous offrent les automates et les programmes dont on peut les munir.

Nous allons donc décrire le traitement possible du texte par un automate, en énumérant un certain nombre de programmes, en respectant un certain enchaînement logique, mais sans insister sur les conditions rigoureuses (format d'entrée et de sortie) qu'exigerait l'enchaînement effectif dans un automate réel.

Rappelons que ces programmes mettent en oeuvre

- un algorithme général (tel l'algorithme d'analyse syntaxique) ;
- une documentation de base (telle que le dictionnaire) ;
- des données particulières (le texte à traiter) ;
- un résultat.

Nous avons attribué à chacun d'eux un symbole mnémonique comme pour les chapitres précédents.

ANALYSE LEXICALE

Cette analyse, quoique triviale est évidemment fondamentale. D'ailleurs la lexicographie se tourne désormais de plus en plus vers l'utilisation des machines qui lui sont indispensables lorsqu'elle se propose de manipuler des masses vraiment significatives de textes (i).

IL	algorithme : comparaison ;
Identification lexicale	documentation de base : lexique ;
	données : les mots.

(1) On se reportera notamment avec fruit aux travaux de j. Quemada à Besançon.

Il s'agit simplement de vérifier que les mots utilisés existent (dans le lexique de base utilisé). Par exemple, si ce lexique est le *Petit Larousse illustré,* les mots « z' » et « ânée » seront indiqués comme aberrants. Nous n'indiquerons pas ici comment ces anomalies pourraient être prises en charge par un programme spécialisé (dont la documentation de base serait de nature phonétique).

CL — algorithme : comptage ;
Comptage — documentation de base : programme A,;
lexical — données : les mots.

Il s'agit d'attribuer à chaque mot identifié un registre qui augmente d'une unité chaque fois qu'on rencontre à nouveau le même mot. L'application de A_1 - A_2 au texte choisi T donne la statistique suivante (chaque mot est suivi du nombre de ses apparitions dans le texte

le	16	des	6	ce	4			
la	15	d'	6	ces	4			
de	12	Saturnin	6	dans	4			
à	11	au	5	garçon	4			
1'	9	est	5	seulement	4			
une	8	il	5	etc.				
un	8	les	4					
les	7							

L'examen, même rapide, de cette statistique en montre les limitations « de» et « d' », « se » et « s' », etc., sont comptés comme des mots différents. Il en va de même de : « est », « être », « furent », etc., ou de « fille » et « filles ». Par contre, on a identifié des apparitions de « la », « leur », etc., qui ne correspondent pas au même être linguistique.

Parmi les programmes de regroupement, nous pouvons définir dès maintenant

RL — algorithme : élimination des élisions ;
Regroupement — documentation de base : liste de règles de morphophonémique ;
lexical — données : mots.

La mise en aeuvre de RL sur T fera d'ailleurs sortir de la liste des mots identifiés, le « c' » de « ç't'ânée » auquel ne correspond aucune élision visible.

Mais bien entendu le programme de regroupement le plus important est le suivant

RP — algorithme : identification de mots qui appartiennent à un même paradigme grammatical ;
Regroupement — documentation de base
paradigmatique — - règles des genres et des nombres ;
- listes de pluriels irréguliers, tables de conjugaisons (régulières ou non) ;
données : mots.

On conviendra de remplacer les formes fléchies par la forme singulier masculin ou la forme infinitive, suivant la nature grammaticale des mots.

Après insertion de RL-RP entre IL **et** CL, on obtient la nouvelle statistique

le	40	ce	12					
de	24	être	9					
à	16	Saturnin	6					
un	16	etc.						

Le document de base qui permet la mise en aeuvre du programme RP peut servir également à produire des couples (mots, nature paradigmatique) tel que

imitent : 3e personne du pluriel de l'indicatif présent du verbe imiter.

Plus généralement, nous utiliserons le programme d'identification grammaticale

IG — algorithme : recherche de la catégorie grammaticale et, le cas échéant, identification du paradigme ;
documentation de base : dictionnaire grammatical ;
données : mots ;
résultat : couples (mot, catégorie ou paradigme).

On notera que le résultat n'est pas nécessairement unique.

C'est ainsi que le quatrième mot de la douzième phrase : « revenu » introduit dans le programme IG produira deux résultats différents (revenu, substantif masculin singulier) et (revenu, verbe revenir participe passé masculin singulier).

Lorsqu'un homme - ou même un enfant - effectue ce genre d'analyse, il élimine automatiquement les possibilités visiblement contra-

dictoires avec le reste du texte. L'automate ne le fait pas, ou plutôt il ne peut le faire qu'après la mise en oeuvre d'autres programmes spécialisés.

ANALYSE SYNTAXIQUE

Ceci veut dire que l'application du programme IG a pour effet principal de rendre manifestes les ambiguïtés du langage. Ces ambiguïtés ne peuvent pas être levées au niveau de la grammaire (au sens usuel du terme, c'est-à-dire déclinaison, conjugaison, etc.) ; il faut faire appel à la syntaxe. Pour bien marquer ce qui fait l'importance et la force de la syntaxe, considérée en tant que système de contraintes, nous utiliserons une *représentation graphique* des contraintes ainsi manifestées. Ce type de représentation est maintenant fort à la mode et on en possède plusieurs systèmes, plus ou moins équivalents.

Le choix, important pour les linguistes, n'est pas significatif en ce qui concerne notre présent effort, qui est seulement un effort d'illustration des méthodes. Aussi, le choix que nous proposons n'est-il nullement impératif. Nous utilisons la méthode due à L. Tesnière qui a, en particulier pour nos lecteurs, le mérite d'être exposé dans un ouvrage rédigé en français (i).

Il s'agit de marquer la contrainte syntaxique qui lie les éléments d'un syngtame en liant physiquement les mots par un trait. Le graphe ainsi obtenu s'appelle « stemma ». Les stemmas dénotent une sorte de hiérarchie engendrée par une hiérarchie des catégories grammaticales, hiérarchie que l'on définit par les inégalités

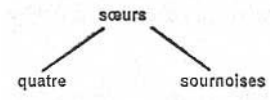
$$|\text{verbe}| > |\text{substantif}| > |\text{adjectif}| > |\text{adverbe}|$$

Des règles complémentaires permettent de relier entre elles des propositions élémentaires dont l'articulation est du type : proposition relative, subordonnée, etc.

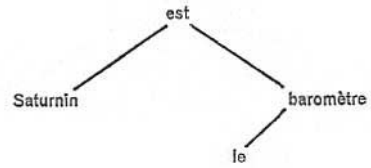
Dans un tel système syntaxique, les lois ont l'apparence des formules de la chimie organique

(1) L. TESNIÈRF, *Éléments de syntaxe structurale,* Klincksieck, 1959. De nombreux auteurs utilisent maintenant la notation de Tesnièrc, sans toujours le mentionner !

C'est ainsi qu'au début de la première phrase la suite « quatre sueurs sournoises » donne



Phrase 6, après la virgule, la suite « Saturnin est le baromètre » donne



Ces formules pseudo-chimiques n'expriment rien d'autre, bien entendu, que les règles classiques de la grammaire, puisqu'elles visualisent la construction progressive de groupes nominaux, de groupes verbaux, etc. Elles permettent, en particulier de prendre aisément en charge les règles d'accord en genre et en nombre, l'articulation des propositions de différents types, etc.

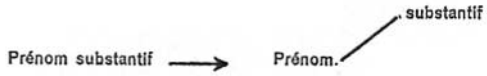
Nous ajouterons donc à la liste de nos programmes

CS algorithme : construction des stemmas relatifs à un groupement de mots donnés ;
documentation de base : syntaxe ;
données : groupes de couples (mot, catégorie grammaticale) ;
résultat : stemmas.

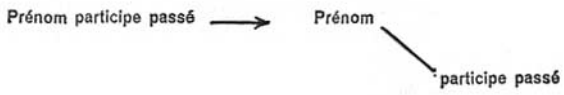
Succès ou échecs de la construction aboutissent à conserver ou à éliminer certains des couples (mot, catégorie) ; on peut donc construire le programme d'élimination des ambiguïtés

EA algorithme : élimination de couples (mot, catégorie) ;
documentation de base : IG CS ;
données : liste de couples ;
résultat : liste restreinte de couples.

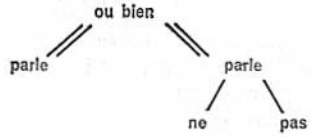
C'est ainsi que le fragment « Saturnin revenu », tiré de la phrase 12, étant compris entre deux virgules, doit être connexe. Or il n'existe aucune règle de grammaire permettant d'écrire



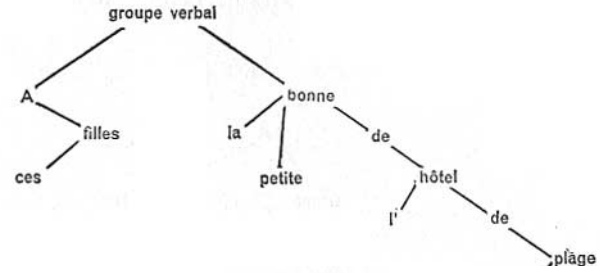
par contre il existe une règle qui donne



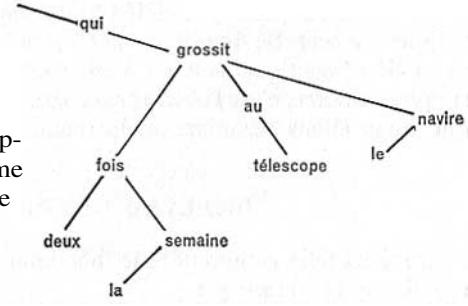
On élimine ainsi l'ambiguïté relative au mot « revenu ».
On imagine sans peine qu'une syntaxe efficace doit comporter un très grand nombre de telles règles dont certaines très rarement usitées mais tout de même indispensables à la cohérence de la langue en tant que système. L'ouvrage de L. Tesnière ne donne que quelques-unes de ces règles et, à vrai dire, on ne dispose, en ce moment, pour aucune langue, d'un ensemble de règles suffisamment proche de l'exhaustivité. Cela veut dire que dans certains cas l'analyse syntaxique ne pourra être menée jusqu'au bout ou que, au contraire, plusieurs constructions seront proposées pour un même fragment.
Pour l'exemple que nous avons choisi, aucune de ces difficultés n'apparaît et chaque phrase donne naissance à un ou plusieurs stemmas. Nous donnons ci-dessous quelques exemples (la numérotation est celle des phrases (de 1 à 20)
Examinons tout d'abord la *phrase 4* qui présente plusieurs particularités intéressantes
Tout d'abord le « sommet » du graphe n'est pas un mot, mais un groupe de mots



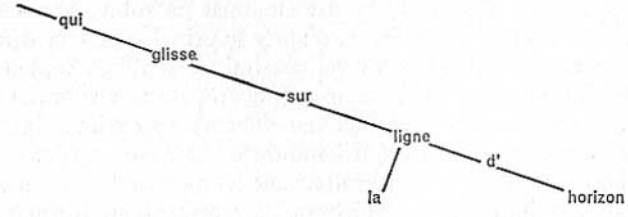
A ce groupe sont rattachés un graphe sujet et un graphe complément, mais avec une inversion



le substantif « bonne » reçoit un qualificatif supplémentaire sous la forme d'une proposition relative

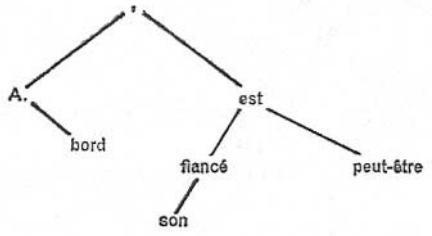


le mot « navire » à son tour est qualifié à l'aide d'une proposition relative



d'où finalement une structure globale extrêmement télescopique.

Un autre exemple intéressant est celui de la *phrase 5* où l'expression « à bord », qui qualifie le verbe « est », ne peut être acceptée *avant* le verbe que grâce à la présence de la virgule, ce qui donne le graphe



Mais cette phrase présente encore un autre intérêt : c'est la présence de l'adjectif possessif « son » que notre construction syntaxique permet de lier à « fiancé » mais qui se trouve évidemment lié sémantiquement à « bonne ». Établir de telles connexions supplémentaires, c'est l'objet des programmes d'analyse sémantique, que nous allons examiner maintenant.

ANALYSE SEMANTIQUE

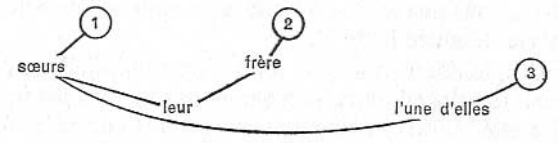
Pour mieux faire comprendre le fonctionnement de ces programmes, examinons la phrase 4

« A ces filles parle ou bien ne parle pas, pour des raisons ignorées, la petite bonne de l'hôtel de la plage qui, deux fois la semaine, grossit au télescope le navire qui glisse sur la ligne d'horizon. »

De toute évidence, la structure des stemmas ne suffit pas à épuiser la cohérence du texte. D'ailleurs, d'après le principe de construction, les sommets sont toujours des verbes, qui ne sont pas toujours les mots les plus significatifs en stemma (notamment s'ils sont des auxiliaires). Une première retouche au schéma syntaxique est alors apportée par la construction (déjà indiquée par Tesnière) des liaisons « anaphoriques », c'est-à-dire rattachant les pronoms personnels ou possessifs aux individus ou objets qu'ils représentent. Il s'agit donc d'un nouveau programme :

CA algorithme : tracé de nouvelles connexions entre mots ;
documentation de base : loi des connexions anaphoriques ;
données : phrases du texte ;
résultat : liens horizontaux entre stemmas.

Appliqué au texte T, CA donne, par exemple, le résultat suivant



La simple application des connexions anaphoriques ne suffit pas à expliciter toutes les connexions du même type. C'est ainsi que « ces », de « ces filles », et aussi de « ces êtres », réclame une connexion que les règles élémentaires ne donnent pas.

Il faut donc mettre en oeuvre un nouveau programme qui va plus profondément dans l'analyse sémantique, par exemple

CH algorithme : mise à jour de connexions hiérarchiques ;
documentation de base : classifications diverses ;
données : mots ;
résultats : nouveaux liens anaphoriques.

Dans les classifications en question, on voit notamment que les soeurs sont nécessairement des filles, que les filles, bonne et fiancé, sont des êtres, etc.

Remarquons que, sauf les exceptions des classifications des espèces vivantes, ou des minéraux, des composés chimiques, etc., de telles classifications ne sont pas, pour l'instant, complètes.

Enfin, de nouvelles connexions plus générales, de type sémantique, peuvent apparaître, telle celle qui, dans les phrases 4 et 5, relie « navire » et « à bord ». Elles seront mises en évidence à l'aide d'un programme dont les documents de base pourraient être des ouvrages du type « recueil de mots français par catégories s, dictionnaires analogiques, etc., d'où le programme qui suit.

CS algorithme : établissement de connexions sémantiques ;
documentation de base : liste de mots classés par sujet ;
données : mots ;
résultat : nouvelles connexions entre stemmas.

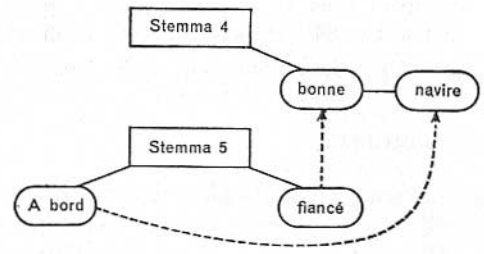
De nouveaux programmes sémantiques plus complexes ou plus abstraits doivent être également mis en oeuvre. Pour ne pas surcharger notre analyse, nous ne citerons ici que ceux dont la nécessité s'impose pour l'analyse de notre texte T.
Remarquons que désormais nous nous situons au-delà de ce qui est effectivement développé. De tels programmes ne sont pas impossibles à construire mais coûtent évidemment un effort considérable qui n'a été qu'abordé dans les différentes équipes qui travaillent aujourd'hui dans les divers domaines de la linguistique automatique.

ST algorithme : établissement de connexions spatiales ou temporelles ;
documentation de base : règles de grammaire sémantique spéciales (relatives notamment aux aspects du verbe) ;
éléments : mots ou syntagmes ;
résultat : nouvelles connexions entre polystemmas.

Cet algorithme met en évidence des relations de voisinage, de succession, etc. Il doit être distingué de CH en raison de l'utilisation de règles particulières
Ainsi, le « puis » de i8 assure la liaison avec le groupe 16-17. Les passés simples « furent » et « repartirent » unissent i8 et i9. Par contre, le présent « aiment » de 2o affirme la séparation temporelle de 2o d'avec les groupes précédents.
Tout ceci nous permet donc d'achever d'établir la connexion globale de l'ensemble du texte, ainsi d'ailleurs que le manifestera plus clairement encore le programme suivant

CG algorithme : substitution aux deux stemmas d'un symbole (par ex. le numéro du stemma) auquel se rattachent les mots reliés anaphoriquement, puis fabrication d'un graphe où ces liaisons, ainsi que les autres liaisons sémantiques qui viennent d'être définies sont explicitées ;
documentation de base : l'ensemble des programmes précédents ;
données : texte complet ;
résultat : graphe récapitulatif des connexions textuelles.

Appliqué à T, CG donne le résultat suivant



Dans de nombreux cas, l'établissement du métastemma permettra de résoudre les tâches pour lesquelles un automate avait été nécessaire.

Pourtant, lorsqu'on veut aller plus loin dans la compréhension, une deuxième phase de l'analyse peut être nécessaire. Cette phase que nous allons décrire maintenant, en utilisant toujours notre texte T comme illustration, nous fera passer du métastemma au diagramme, en passant par l'intermédiaire d'une formalisation.

Ici encore, les diverses étapes seront présentées en termes de programmes exécutables par un automate. Nous utiliserons successivement

II Identification des individus
algorithme : comparaison ;
documentation de base : listes d'individus ;
données : noms propres, noms de personnes, etc.
résultat : couples individus-symboles.

II (T) -* quatre so.urs, un fiancé, trois garçons, etc.

ID Identification des décors
algorithme : comparaison ;
documentation de base : listes des décors, lieux, etc. ;
données : syntagmes ;
résultat : listes de couples (figure, symbole).

ID (T) -* bord de la mer, falaise bistrot, etc.

IO Identification des objets
algorithme : comparaison ;
documentation de base : listes d'objets ;
données : mots ou groupes de mots ;
résultat : couples (groupes de mots, symboles).

10 (T) -> pierre, télescope, navire, etc.

et de même on construira

un programme IT pour l'identification des instants
— IE — événements
— IP — prédicats élémentaires
— IR — relations

puis un programme CP utilisant les résultats des programmes précédents, ainsi que les programmes construits dans le cadre de la logique des prédicats (au sens du chapitre précédent), pour construire des prédicats complexes.

L'ensemble de ces programmes d'identification appliqué à T donne le résultat suivant

a) Individus

I_1 : sueurs
I_2 : garçons
I_3 : frère
I_4 : garçons
I_5 : filles
I_6 : bonne
I_7 : fiancé
I_8 : Saturnin
I_9 : jardinier
I_{10} : Giraudoux
I_{11} : marin
I_{12} : sueur
I_{13} : fille
I_{14} : garçons
I_{15} : garde-champ.
I_{16} : Pierrot
I_{17} : mère
I_{18} : curé
I_{19} : couple
I_{20} : championne
I_{21} : admirateur
I_{22} : estivants
I_{23} : gens
I_{24} : amoureux
I_{25} : huître

b) Prédicats

P_1 ($\equiv (I_1)) = 4$
P_2 (I_1) = sournoises

P_3 sera défini lors de l'analyse de la phrase n° r
P_4 (I_1) = frère
P_5 (I_3) = pêcher
P_6 (I_1) ($\equiv$ E) = l'une d'elles – I_{26}
P_7 (I_{26}) = être vu
P_8 (I_{26}) = tenir (un concile)
P_9 = être assis

c) Actions

A_1 : se retourner
A_2 : oser
A_3 : imiter
A_4 : pêcher
A_5 : voir
A_6 : tenir
A_7 : parler
A_8 : grossir
A_9 : glisser
A_{10} : bricoler
A_{11} : chasser
A_{12} : aller
A_{13} : dormir
A_{14} : découcher
A_{15} : porter
A_{16} : dire
A_{17} : rétablir
A_{18} : simplifier
A_{19} : falloir
A_{20} : tenir compte
A_{21} : soumettre
A_{22} : savoir
A_{23} : parvenir
A_{24} : (faire) entrer
A_{25} : accomplir
A_{26} : pleuvoir
A_{27} : se réfugier
A_{28} : regarder
A_{29} : reconnaître
A_{30} : aller
A_{31} : vouloir
A_{32} : faire un discours
A_{33} : concerner
A_{34} : permettre
A_{35} : repartir
A_{36} : aimer
A_{37} : refermer
A_{38} : régner

d) Événements :

E_1 : houle
E_2 : effet
E_3 : mot
E_4 : concile
E_5 : ordre social
E_6 : retour

e) Lieux :

L_1 : Terre-Neuve
L_2 : à mi-falaise
L_3 : entre des haies
L_4 : bistrots
L_5 : chez sa sueur

f) Objets

O_1 : mer
O_2 : hanche
O_3 : baromètre
O_4 : croûte
O_5 : verre
O_6 : balai
O_7 : horloge

g) Modes :

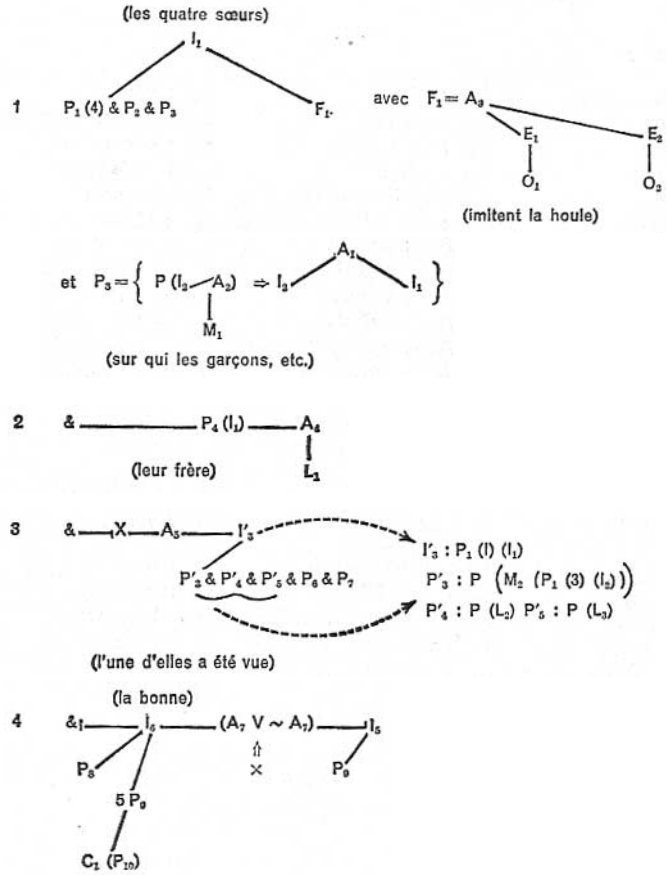
M_1 seulement
M_2 : avec
M_3 : comme
M_4 : pour

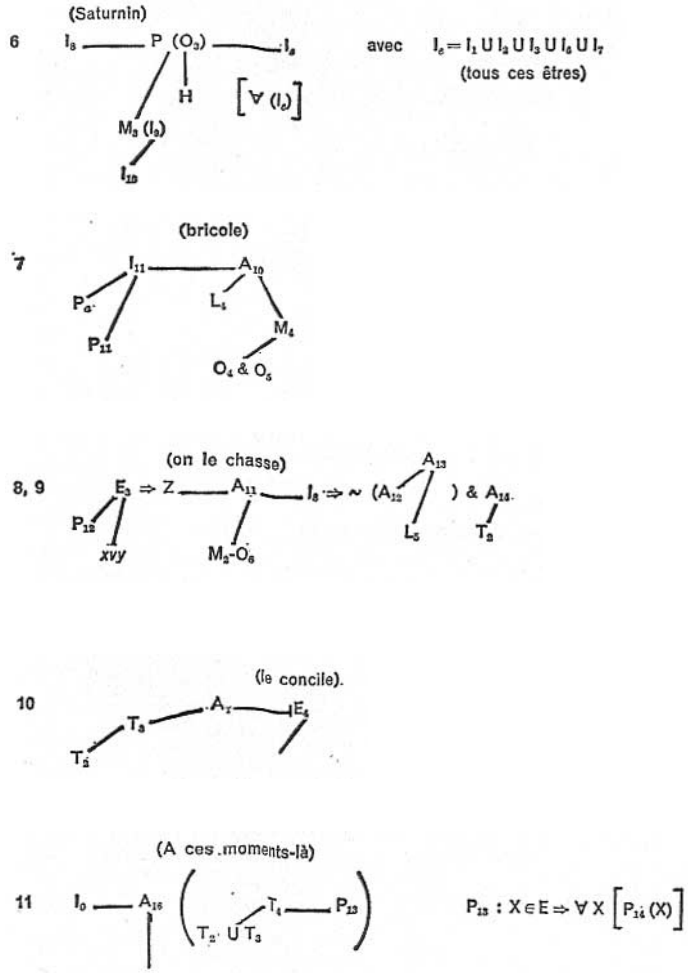
h) Temps

T_1 : un soir de l'été
T_2 pendant des jours

CONSTRUCTION DE DIAGRAMMES

En appliquant le programme CP, on obtient alors une représentation de T sous forme d'une proposition qui se décompose en 20 prédicats reliés par la conjonction &, de la façon suivante





12 (Saturnin revenu)

$P_{15}(I_4) \Rightarrow P_{14}(E_6)$

$M_3 — E_6 — O_7$

$M(E_7) \quad P_{17} \& P_{18}$

(ce n'est pas un fait de nature à ...)

13 $C_1\left[\sim\left[P_{19} \quad [P_7]\right]\right] \& P_{21}$ avec $P_7 = P_{11} \& P_{12}$

$P_{20} — A_{17} — N_1$

$P_{21} : X \in 5 \Rightarrow \forall_x \left(A_{16}(x) — A_{19}(\dot{P}_7) \vee A_{20}(P_7) \right)$

$M_8(E_7)$

(le garde-champêtre le sait) (Pierrot le facteur)

14 $I_{15} — A_{21} — P_{22}$ & $I_{16} — P_{23}$

(Monsieur le curé)

$\& — (I_{17} \& I_{18}) — P_7$

$C \quad I_{16} \quad \sim P_{22}(P_{23})$

$T_3 — E_7$

$P_{25} \& P_{26}$

avec $P_{26} = I_{15} — A_{24}$

$L_5 \quad P_{27} \quad T_6$

$P_{25} : P_{21} — A_{23} — N_2$

P_{24}

(la dernière fois que j'ai vu Saturnin)

15 $(I_0 — A_5 — I_8) — (E_9 \& I_8 — A_{29} \& I_{10} — A_{26})$

$T_7 \quad P_{28} \quad P_{29} \quad M_7 \quad L_6$

(il fit un discours de biais)

16

$T_7 \quad (I_8 — A_5 — I_{19}) \Rightarrow I_{19} — A_{10} — i_{19}$

$\sim A_{29} \; P_{30} \quad E_9 \quad M_{10}$

$\& — M_{11} \; P_{31}(E_9)$

$P_{32} \Rightarrow P_{33}$

(« Z'ont point d'chance ... »)

17 $D(I_9, I_0) \equiv N \sim \left[I_{22} \; P_{33}\right]$

T_8

(Puis ce furent)

18, 19 $T_8(E_{10} \& I_{23} — A_{13})$

$P_{34} \quad L$

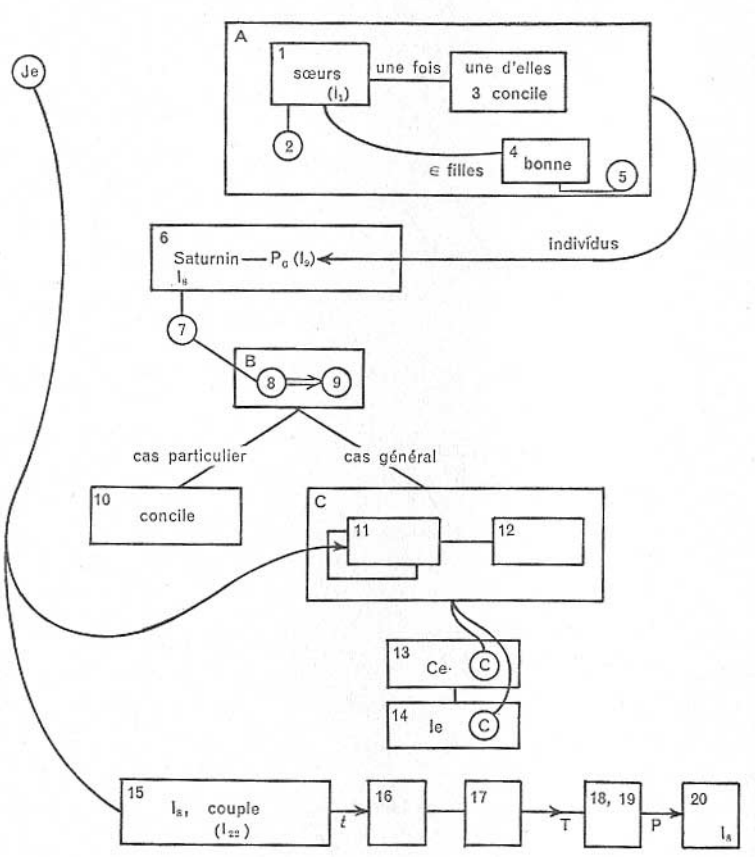
(les amoureux de six heures)

20 $T_9(I_{21} — A_{14})$ & $L — A_{15}$

$T_{10} \quad M \quad C \quad T_{11} — I_{15} — A_{16}$

C

Une représentation plus concentrée du texte s'obtiendra alors facilement (en regroupant convenablement certains prédicats)



EXPLOITATION DES ANALYSES

Les analyses lexicales sont exploitables de bien des points de vue. Par exemple, les premiers essais de traduction automatique faisaient usage d'une telle analyse, suivie de l'utilisation de dictionnaires simples. Il s'agissait par conséquent d'un simple mot à mot (i). On utilise aussi ces analyses pour l'établissement de *concordances* qui établissent l'existence de rapprochement plus ou moins fréquents entre certains mots, enfin pour l'analyse stylistique (2).

Les analyses syntaxiques forment l'essentiel du domaine actuel de la linguistique automatique. De nombreuses équipes aux U.S.A., en U.R.S.S., en France, en Italie, en Angleterre, etc., se sont attaquées à ce problème. Dans la plupart des cas l'objectif à atteindre était la traduction automatique. Dans certains cas, cependant, l'analyse syntaxique était un but en soi (par exemple dans les études menées sous la direction de Z. Harris à l'Université de Pennsylvanie)

L'expérience a montré que la mise au point de grammaires réellement complètes était un travail immense et, jusqu'à présent, inachevé. Mentionnons cependant les progrès réalisés par le Japonais Kubo à l'Université Harvard (il s'agit de l'explicitation - à l'intention d'un automate - de la grammaire anglaise ; les grammairiens prétendent souvent que la grammaire anglaise tiendrait en quelques pages ; cependant les règles mises à jour par Kubo sont au nombre de plusieurs milliers).

Le niveau sémantique a été abordé suivant des voies assez diverses. Les uns, comme S. Ceccato, à l'Université de Milan, n'ont pas voulu séparer l'aspect syntaxique et l'aspect sémantique dans l'analyse du langage. Leur intention, plus psychologique que linguistique, se manifeste bien par le nom *Adamo II* qu'ils avaient donné à leur projet d'automate.

L'attaque la plus massive sur le front sémantique est celle qu'ont menée les documentalistes. Leur but était double. Dans une première phase, ils voulaient définir un langage artificiel (artificiel par sa gram-

(r) Voir par exemple le livre d'E. Delavenney cité en bibliographie.
(2) On trouvera de nombreuses illustrations de ceci dans les ouvrages de P. Gurxaun. Voir aussi J. Cohen, *Structure du langage poétique,* Flammarion, 1966.

maire, mais naturel par son vocabulaire) qui pût décrire les documents (et aussi les questions auxquelles ces documents devraient fournir une réponse). Dans une seconde phase ils voulaient construire un programme qui donne à un automate la possibilité, à la lecture du document (ou au moins de son résumé) de construire la phrase ou les phrases qui, dans le langage artificiel ainsi défini, en exprimaient le contenu.

Dans une première étape les langages artificiels furent totalement dépourvus de grammaire. Ils ne comportaient que des *mots clés. Puis* on introduisit des relations qui permirent la constitution de *phrases clés (1).*

Mais la vraie difficulté n'était pas dans la constitution d'une syntaxe artificielle, mais dans la tentative d'exprimer, d'une façon ou d'une autre, la signification et les rapports de signification.

De nombreux essais furent tentés dont plusieurs sont maintenant en application (2).

D'autres essais intéressants sont ceux qui se sont efforcés de manier des fragments très spécialisés du langage naturel, de façon à réduire autant que possible l'étendue du champ sémantique correspondant. On en a vu un exemple dans l'article de Slagle que nous avons cité, à titre d'exemple, dans notre introduction.

Dans tous les cas les travaux en cours sont loin d'être suffisants pour qu'on dispose dès à présent d'un réseau de relations sémantiques vraiment efficace. On peut cependant penser que si les recherches se poursuivent la situation ira en s'améliorant régulièrement.

(1) P. BRAFFORT et A. LETTOV, Des mots clés aux phrases clés, *Bulletin des bibliothèques de France, 1959, P. 383.*

(2) Voir l'article de C. WASTON, Information retrieval, dans *Advances in Computers, 5, 1965.*

CHAPITRE VI

La complexité

Simulation du joueur, du mathématicien (dans certaines de ses activités les plus simples), du sujet parlant (avec également des restrictions), au cours des trois chapitres précédents, nous avons montré comment les automates pouvaient prendre en charge la résolution de problèmes de plus en plus difficiles et que l'on situe ordinairement dans le domaine des activités intellectuelles.

Parvenus à ce point, avons-nous le droit de dire que nous avons effectivement démontré l'existence et l'efficacité d'une « intelligence artificielle » ? Pas exactement : car nous avons pris le soin, pour tous les cas traités, de donner une analyse assez détaillée des programmes que les automates devraient exécuter ; mais, dans la plupart des cas, nous n'avons pas discuté de ce que représentaient quantitativement ces programmes du point de vue de la capacité et de la vitesse dont ces automates devraient être pourvus. Dans plusieurs cas il s'agissait d'ailleurs de schémas de programmes, et non pas de programmes effectivement écrits.

Cela veut dire, certes, que l'intelligence artificielle est possible *en principe.* Et puisque, à plusieurs reprises, nous avons fait allusion à des expériences déjà effectuées (jeu de Go-Bang, démonstrations de propositions logiques, traduction du russe vers l'anglais), nous savons même que ce principe est devenu, en diverses occasions, un *fait.* Mais à aucun moment nous n'avons touché du doigt les *limites* de ce qui est effectivement possible (de ce qui est possible aujourd'hui comme de ce qu'il est raisonnable de prévoir pour demain).

Nous sommes évidemment arrivés au point où une telle estimation devient indispensable. Or, on va voir qu'en s'y attelant on met en valeur une notion qui est commune à toutes les rubriques que nous avons successivement abordées et qui explicite vraiment l'unité du domaine de l'intelligence artificielle : il s'agit de la notion de *complexité.*

A plusieurs reprises nous avons indiqué, soit explicitement, soit implicitement que, dès qu'un problème était complètement formulé - c'est-à-dire dès qu'on disposait d'un système formel convenable dans lequel le problème en question prenait l'aspect d'une formule à découvrir -, on pouvait définir un automate et un programme pour effectuer le travail.

En réalité cette affirmation demanderait à être précisée et complétée car certains énoncés mathématiques (qui appartiennent à ce que l'on appelle le domaine *non constructif)* posent ici des difficultés particulières.

Mais si nous nous bornons (ce qui est parfaitement licite pour les domaines que nous avons abordés) au domaine *constructif,* nous pouvons nous en tenir à ce point de vue qui consiste, en somme, à associer à chaque problème une machine de Turing particulière, pour reprendre le modèle d'automate que nous avons présenté dans le chapitre II.

Nous avions alors souligné que si la machine définie (dès 1936) par Turing n'était guère semblable aux calculatrices électroniques que nous connaissons, rien ne s'opposait à ce que l'on programme une calculatrice moderne pour la faire se comporter comme une machine de Turing si ce n'est le point fondamental suivant : la longueur de la bande utilisée par la machine de Turing pour effectuer un calcul est *illimitée* alors qu'il ne peut évidemment pas en être ainsi d'une machine réelle quelle que soit la nature de la mémoire utilisée pour simuler la « bande» de la machine de Turing.

Et cette remarque fondamentale, c'est au fond la rencontre initiale de ce *mur de la complexité* auquel nous avons fait allusion ici et là, au cours de cet ouvrage.

L'EXPLOSION COMBINATOIRE

« Complexe » se situe quelque part entre «possible » et «impossible», mais c'est « impossible» qui demande tout d'abord à être précisé. Prenons pour cela deux exemples

a) Soit un alphabet formé de lettres *a, b, c...* Nous voulons ranger des mots composés avec des lettres de cet alphabet, dans l'ordre « lexicographique ». Si la longueur des mots que l'on veut classer n'est pas limitée *a priori,* la capacité de l'automate qui effectuera le classement doit être également illimitée. Car si un mot de longueur *n* (très grand) est déjà classé et que survient un second mot de longueur au moins égale à *n,* il est possible que les lettres d'ordre r, *2, ..., n* -1 soient identiques et que la décision n'intervienne que pour la dernière lettre. Il faut donc stocker n lettres en mémoire et effectuer *n* comparaisons. Si *n* est illimité, il en va de même de la capacité qu'on exigera de l'automate, et du temps pendant lequel il devra être utilisé. Cette remarque vaut en ce qui concerne le classement respectif de deux mots. Si l'on veut maintenant insérer un mot dans une liste de mots déjà classés, il faudra répéter cette opération un certain nombre de fois (les mots déjà classés étant stockés en mémoire). S'il y a N mots à classer, de longueur maximum *n,* un calcul simple montre qu'il pourrait être nécessaire d'effectuer

$$n. \sum_{1}^{N-1} m = n \frac{(N-1)(N-2)}{2}$$

opérations élémentaires de comparaison.

Supposons que *n* soit de l'ordre de 10^3 et N de l'ordre de 10^5, le nombre d'opérations sera de l'ordre de 10^{13}, ce qui est en dehors des possibilités des machines actuelles (ou envisageable dans un proche futur), car il faudrait faire travailler une telle machine pendant une année entière sur le même problème de classement.

b) Reprenons le jeu de Go-Bang étudié à la fin du chapitre III. Sur une grille composée de 19 lignes et 19 colonnes, il y a 361 positions. Le joueur qui commence peut donc choisir entre 361 possibilités ; le suivant entre 360. Le premier joue à nouveau et a 359 Possibilités, etc. Si la partie s'arrête au bout de n coups (en totalisant le

nombre des coups joués par chacun des adversaires), le nombre de combinaisons possibles est évidemment

$361.360 \ldots\ldots (361 - n)$ c'est-à-dire $\dfrac{361!}{(361-n-1)!}$

Une partie normale ne dépasse guère une cinquantaine de coups, mais cela donne déjà un nombre de combinaisons qui excède largement 10^{100}

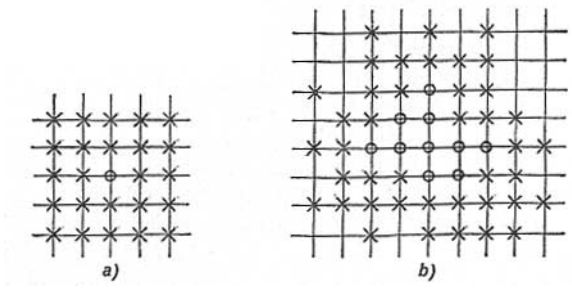


FIG. 31. — Coups envisagés
a) En début de partie ; b) au cours d'une partie

Bien entendu les joueurs sont loin d'exploiter ces possibilités. Le premier place son pion au voisinage du centre de la grille. Le second riposte en se plaçant non loin du premier, de façon à gêner d'éventuelles combinaisons. Supposons alors que chaque joueur n'ait à choisir que parmi les points de la grille qui ne sont pas distants de plus de deux noeuds d'un pion déjà posé (voir fig. *31).*
Il est facile de voir sur la figure (et on peut s'en convaincre plus complètement par le calcul) que le nombre des choix possibles reste supérieur à 20 après chaque coup (et augmente en général au fur et à mesure que la partie se développe). Dans ces conditions les combinaisons possibles, pour une partie d'une cinquantaine de coups, seraient certainement en nombre supérieur à 10^{50}, ce qui reste très au-delà des possibilités d'exploration par un automate.

c) A l'occasion de notre brève introduction à la théorie de la déductibilité, présentée au cours du chapitre IV, nous avons rencontré deux types de fonctions, celles qui appliquent un ensemble E dans

lui-même, et celles qui appliquent l'ensemble E dans l'ensemble B des « valeurs de vérité », ensemble qui comprend seulement deux éléments : { vrai } et { faux }. Si l'on suppose que l'ensemble E contient *n* éléments et que les fonctions sont à *k* arguments, on peut montrer que le nombre de fonctions de E dans E différents est égal à n^{nk}, celui des fonctions de E dans B étant 2^{n^k}.
Pour fixer les idées, on peut voir que pour E réduit à trois éléments et $k = 2$, le nombre des fonctions du premier type est déjà *19 683* et celui du second type 512. Or il est clair que pour les recherches pratiques de démonstration de formules tant soit peu intéressantes on doit utiliser des ensembles E beaucoup plus grands et des fonctions de plus de deux arguments.

Il est donc clair qu'ici encore on tombe très vite dans un domaine complètement inaccessible aux automates réels.

Il n'y aurait aucune difficulté, on s'en doute, à développer des exemples du même type pour les problèmes liés au traitement automatique du langage naturel : tous ces exemples sont des manifestations de ce que l'on appelle souvent *l'explosion combinatoire.* Cette expression rend, sous forme imagée, l'impression que l'on ressent lorsqu'on se livre, pour ces divers problèmes, à des calculs d'ordres de grandeur.

Ce qui caractérise, en effet, de tels ordres de grandeur, c'est la *vitesse* avec laquelle se développe l'ampleur du problème lorsque croît le nombre des éléments qui le caractérisent. Pour les problèmes traditionnels, la dépendance est linéaire ou quadratique, c'est-à-dire que lorsque le nombre des éléments croît comme *n,* la complexité du problème est proportionnelle à *n ou* au carré de n.

Mais dans les problèmes qui nous intéressent ici, la situation est bien différente : les ordres de croissances sont du type 2^n ou n^n ou pire encore ; la croissance est *exponentielle !*
Dans ces problèmes en effet, si on veut se livrer à un examen exhaustif des cas possibles, on doit utiliser, pour dénombrer ces cas, les formules de *l'analyse combinatoire* (élevée au rang de discipline mathématique par Leibniz), formules qui servent à évaluer le nombre de combinaisons de *n* objets soumis, dans leur arrangement, à des conditions de divers types.

C'est ainsi que si l'on se donne n objets différents et qu'on les dispose en file indienne, le nombre total des files distinctes que l'on peut ainsi constituer (que l'on appelle nombre des *permutations* de la collection) est égal *à n! (1). Or n!* est une fonction de n qui varie très rapidement lorsque n augmente. On en a une idée en utilisant la formule approchée duc à Stirling

$$n! \sim \sqrt{2\pi} n^{n+1/2}$$

pour $n > 10$ $n!$ devient très vite « astronomique ».

Les remarques qui précèdent permettent de comprendre cette expression « explosion combinatoire ». Associées aux exemples qui les ont précédées, elles donnent un aspect plus quantitatif à cette notion de « mur de la complexité » à laquelle il est si fréquent de voir les auteurs se référer.

Cependant il est possible, et utile, d'entrer un peu plus avant dans les détails de quelques problèmes afin de ne pas se contenter de voir naître et croître la complexité, mais de comprendre comment on peut se donner - au moins dans certains cas - les moyens de la *réduire.* C'est ce que nous allons faire à l'aide d'un problème mathématique particulier : le problème des mots.

LE PROBLEME DES "MOTS"

A la fin du chapitre II, nous avons rapidement indiqué comment la notion même d' « écriture », c'est-à-dire du rangement de lettres les unes à côté des autres, pouvait être associée à une structure algébrique particulière, celle de « monoïde ».

Un monoïde engendré par un alphabet, c'est l'ensemble de tous les mots que l'on peut écrire en prenant plusieurs lettres de cet alphabet (avec éventuellement des répétitions).

Soit l'alphabet

$$A=\{a,b,c,d,e\}.$$

(1) Rappelons que n ! (que l'on lit "factorielle n" désigne le produit 1.2.3...(n-1)n.

Les « mots » *aaabb, dbca, b, dbdbbdbd,* etc., font partie du monoïde engendré par A :

Si l'on ne se donne pas d'autre contrainte, le monoïde en question est le monoïde *libre* engendré par A.

Mais on peut s'imposer certaines relations qui *identifient* des mots du monoïde. Il est possible alors de ne considérer qu'un ensemble restreint où, lorsque des mots du monoïde sont identiques d'après l'application des relations d'identification, on les remplace par l'un d'entre eux, ou par un symbole de leur classe.

Par exemple si les relations du monoïde comprennent le groupe :

$$ab \rightleftarrows b \rightleftarrows ba$$
$$ac \rightleftarrows c \rightleftarrows ca$$
$$ad \rightleftarrows d \rightleftarrows da$$
$$ae \rightleftarrows e \rightleftarrows ea$$

tous les mots du monoïde qui contiennent un ou plusieurs a et d'autres lettres peuvent être remplacés par le mot obtenu en effaçant tous les a.

Le *problème des mots* est alors le suivant. Soit un monoïde défini par un alphabet et un groupe de relations. Soit deux mots quelconques dans cet alphabet. Définir une méthode qui, pour *tout* alphabet (fini), *tout* groupe (fini) de relation et *tout* couple de mots, permette de décider si les deux mots sont identiques, après utilisation des règles d'équivalence du monoïde.

Sous cette forme très générale, il est normal de penser que le problème ne peut être résolu.

Pourtant même si l'on *particularise* le problème en se bornant à l'alphabet A défini ci-dessus et en choisissant un groupe de relations bien défini, par exemple :

$$ac \rightleftarrows ca$$
$$ad \rightleftarrows da$$
$$bc \rightleftarrows cb$$
$$bd \rightleftarrows db$$
$$abac \rightleftarrows abacc$$
$$eca \rightleftarrows ae$$
$$edb \rightleftarrows be.$$

un jeune auteur soviétique, Teitsin (1), a pu montrer que le problème du mot est encore indécidable.

Remarquons cependant que le problème ainsi précisé est encore, en fait, une *classe de problèmes* puisque les mots à comparer peuvent être quelconques. En effet il est facile de montrer que le problème peut être résolu pour certains mots particuliers. Par exemple, en se reportant à la liste des substitutions permises, on s'aperçoit que le mot

aaabb

ne peut être obtenu à partir d'aucun autre et ne mène vers aucun autre. Il n'a donc pas d'équivalent.

Au contraire, si on choisit comme couple de mots à comparer l'un de ceux qui servent à définir les relations d'équivalence, par exemple *eca* et *ae,* le problème est résolu immédiatement, cette fois-ci, par l'affirmative.

Il est donc clair que la possibilité ou l'impossibilité de résoudre le problème est liée ici à la quantité d'information qui est contenue dans l'énoncé du problème particulier. C'est un point sur lequel nous allons bientôt revenir.

CALCULABILITES THEORIQUE ET PRATIQUE

Le résultat que nous venons d'évoquer, c'est-à-dire l'impossibilité de construire un algorithme général pour résoudre le problème des mots est un des nombreux *théorèmes de limitation* qui sont apparus, à partir de 1930, sur les frontières des mathématiques et de la logique (2). Les plus célèbres sont : le théorème de Gödel qui conclut à l'impossibilité de démontrer la non-contradiction de l'arithmétique dans un système qui ne soit pas plus fort que l'arithmétique elle-même ; le théorème de Church qui montre l'existence, dans tout système formel comprenant au moins le calcul des prédicats, de propositions indécidables. Les diverses démonstrations de ces théorèmes

(1) Cité par TRAHTENIIROT, *Algorithmes et machines à calculer,* Dunod, 1962.
(2) Sur tout ceci, on consultera avec profit le grand ouvrage de J. LADRIERE, *Les limitations internes des formalismes,* Gauthier-Villars, 1957.

de limitation se rattachent à quelques schèmes fondamentaux sur lesquels nous reviendrons un peu plus loin ; notons dès à présent que l'utilisation de la notion de machines de Turing permet d'énoncer ces faits de limitation sous la forme de l'impossibilité de construire certains programmes.

Dès lors il devient relativement facile de comparer les impossibilités «relatives» évoquées dans le paragraphe consacré à l'explosion combinatoire aux impossibilités « absolues » que nous venons d'évoquer :

- dans certains cas, il est impossible de construire un programme, pour des raisons liées à la nature même du problème : c'est le cas du problème général des mots que nous venons d'évoquer ;
- dans d'autres cas, il est possible de construire un programme, au moins en principe, mais il n'est pas *effectivement possible :*
  - soit de l'écrire complètement ;
  - soit de trouver un automate réel qui possède la capacité de mémoire nécessaire ou qui puisse fonctionner le temps nécessaire à la résolution complète du problème.

A première vue, la distinction semble très importante. En effet, pour ce que l'on sait être impossible *a priori,* rien de raisonnable ne peut être tenté. Par contre, s'il s'agit d'une limitation pratique, on peut s'efforcer de trouver, dans chaque cas, une procédure *ad hoc* qui permette de se tirer d'affaire.

Par exemple, si nous revenons à l'exemple *a)* de la page 119, lorsqu'on veut ranger un mot dans une liste déjà constituée ; on peut se proposer, au lieu d'attaquer les comparaisons par le premier mot qui se présente, commencer au milieu de la liste. La première comparaison élimine ainsi l'une des moitiés de la liste. On se porte alors au milieu de la moitié restante, et ainsi de suite. On peut alors voir que le nombre d'opérations n'augmente pas proportionnellement au *carré* de N, mais au *logarithme* de N. Avec les valeurs numériques que nous avions choisies plus haut, le nombre d'opérations n'est plus de l'ordre de 10^{13}, mais 10^7 et est parfaitement compatible avec les performances des calculatrices actuelles.

Par contre un résultat négatif comme celui relatif au problème des mots semble d'autant plus décourageant que nous nous sommes efforcés, tout au long de cet ouvrage, et en particulier au cours du

chapitre II, de montrer que les problèmes de l'intelligence artificielle pouvaient se mettre sous une forme qui les rende voisins à des problèmes d'identification de mots dans un certain alphabet.

On pourrait donc se demander si les « astuces » qui s'efforcent, dans certains cas particuliers de réduire l'explosion combinatoire ne soit pas un travail de Sisyphe, condamné en fin de compte à l'échec.

C'est donc la possibilité même de l'intelligence qui pourrait être mise en question si cette remarque devait être prise au sérieux.

D'ailleurs il s'est effectivement trouvé un certain nombre d'auteurs, en particulier au début des années 1960, pour contester tout l'effort entrepris et évoquer, à l'appui de leur thèse, les grands théorèmes de limitation de la logique mathématique, et en particulier le théorème de Post. Ce fut le cas notamment de Y. Bar-Hillel et de M. Taube.

Les conséquences de cette offensive se firent surtout sentir dans les domaines de la documentation et de la traduction automatique, mais, d'une façon générale, on put constater un certain recul de l'activité de plusieurs équipes, et même la disparition de certaines d'entre elles.

Aussi nous semble-t-il important d'examiner ce genre d'objections, afin, si possible, d'éliminer les ambiguïtés qui se sont glissées ici et là lorsqu'on se cantonnait dans des généralités au lieu de se livrer à une analyse complète de chaque cas.

Mais nous verrons qu'en avançant dans cette direction il devient possible non seulement d'écarter les objections de principe dont l'importance a d'ailleurs diminué, mais surtout d'ouvrir des perspectives plus générales en ce qui concerne les tactiques à mener dans le combat anticombinatoire.

C'est donc à double titre que la comparaison « calculabilité (ou non-calculabilité) théorique-calculabilité pratique » est importante. Nous allons l'ébaucher en reprenant l'exemple du problème des mots. Pour cela reportons-nous à l'organigramme de la figure 32 qui illustre ce problème sous son aspect « machine ».

On voit que ce schéma ne soulève aucune difficulté dans sa conception. Deux sorties y sont indiquées, une réponse positive et une réponse négative. Mais il est possible également que l'on reste très longtemps dans la boucle sans atteindre une sortie, et ceci parce que les substitutions permises créent constamment des mots nouveaux qui ont

besoin, à leur tour, d'être testés. C'est en ceci que notre programme n'est pas complet : on n'est jamais sûr qu'il s'arrête une *fois* qu'on l'a mis en marche.

Si l'on modifie légèrement le programme, de telle façon qu'un compteur soit défini, qui sera augmenté d'une unité chaque *fois* que la boucle principale sera parcourue une fois, on peut prévoir un test qu'arrête l'automate lorsque le compteur atteint le chiffre K.

Dès lors, quand on se donne deux mots

- ou bien ils apparaissent comme identifiés avant que le compteur atteigne K ;
- ou bien ils apparaissent comme différents avant que le compteur atteigne K ;
- ou bien aucune décision n'est possible en K itérations.

Dans les exemples du paragraphe consacré à l'explosion combinatoire, des calculs élémentaires permettaient de calculer la borne supérieure du nombre des opérations à effectuer. Cette borne était très au-delà de la capacité des automates réels. Cependant on pouvait concevoir qu'une meilleure utilisation des propriétés et des particularités du problème permettrait d'abaisser considérablement la borne supérieure.

Par ailleurs le résultat négatif relatif au problème des mots nous indiquait qu'il n'existait pas de valeur finie de K pour laquelle le problème de l'identification de deux mots *quelconques* a reçu une réponse définitive (qu'elle soit positive ou négative). Mais rien n'empêche d'imaginer qu'en se limitant à la comparaison de mots appartenant à des *classes spécifiques,* correspondant à des propriétés explicites de ces mots, on ne pourrait pas modifier complètement la situation.

De plus, ce qui est important pour justifier les efforts des chercheurs, ce n'est pas tant la borne supérieure, mais la *borne inférieure.* Si nous trouvons que notre problème devra, *dans tous les cas,* nécessiter N opérations élémentaires et que N est trop grand, nous abandonnons. Mais si nous savons seulement que ce nombre *pourrait éventuellement* être voisin de N, nous nous contentons de chercher des procédures qui nous permettent d'éviter cette borne supérieure. Certes,

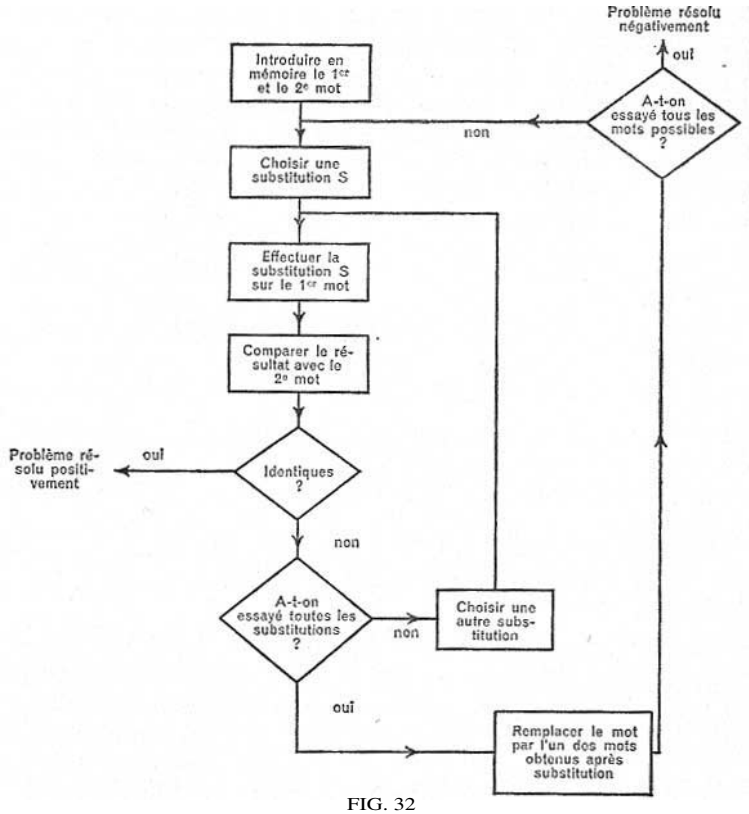


FIG. 32

rien ne nous permet d'être sûr que de telles procédures seront trouvées dans les différents cas où le problème se pose, mais nous sommes désormais dans une situation bien différente de celle qui était évoquée par des auteurs tels que Bar-Hillel ou Taube.

Nous allons illustrer ces réflexions à l'aide de quelques exemples tirés des chapitres précédents.

ANALYSE DE QUELQUES EXEMPLES

Notre premier exemple sera tiré à nouveau de l'ensemble des jeux de grille et nous choisirons, pour simplifier la discussion, le jeu où $a = b = 3$ (c'est-à-dire le Tic-Tac-Dou). Pour rendre les diagrammes plus lisibles, nous remplacerons les noeuds par des cases, d'où le jeu à 9 cases (fig. 33)

1	2	3
4	5	6
7	8	9

FIG. 33. - Le champ du jeu

Au premier coup, 9 choix sont possibles. Le second joueur n'a plus que 8 possibilités, etc.

Au total, on a 9 ! = 9.8.7.6.5.4.3.2 .
= 355 68o possibilités.

Bien entendu, un automate pourrait explorer toutes les possibilités. Mais pour a = 5, on aurait déjà 25! possibilités, ce qui est au-delà de toute possibilité pratique.

Mais ce que nous voulons examiner, dans le cas du Tic-Tac-Dou, c'est que toutes les possibilités ne sont pas également intéressantes.

En effet, un coup d'oeil sur le diagramme ci-dessus montre que, tant qu'il s'agit du 1er coup

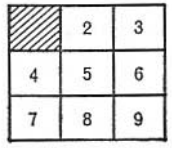
1, 3, 7 et 9 sont équivalents
2, 4, 6, 8 sont équivalents.

Il n'y a donc pas 9 coups différents possibles, mais 3

- un coup du type 1 ;
- un coup du type 2;
- le coup 5.

De même, si le 1er coup joué est 1, pour le second coup

2 et 4 sont équivalents
3 et 7 sont équivalents
8 et 6 sont équivalents (cf. fig. 34).



Fic. 34
Le champ du jeu
après le premier coup

Il y a donc 5 possibilités distinctes

- un coup du type 2 ;
- un coup du type 3 ;
- un coup du type 8 ;
- le coup 5 ;
- le coup 9, etc.

On constate qu'après 2 coups il y a 12 combinaisons réellement distinctes sur les 72 théoriquement possibles. Cela veut dire que les *propriétés caractéristiques du problème que l'on a à résoudre - ici* les propriétés de symétrie de la grille - *réduisent* la multiplication des opérations à effectuer lors du dépouillement de la situation par un automate.
Ceci ne suffit pas, bien entendu, pour assurer la réalisabilité pratique du programme, mais c'est déjà une indication intéressante en ce qui concerne la possibilité de trouver des estimations plus raisonnables de la borne supérieure N.

***

Examinons maintenant le problème d'algèbre élémentaire du chapitre IV.
Il s'agit de vérifier l'identité

$$2m + 2n = 2(m+n)$$

en opérant sur la première formule un certain nombre de substitutions choisies dans la liste : A, S, G, D, C, R, définie page 85.

Il est assez simple ici de vérifier qu'on se trouve dans un cas particulier du problème des mots, tel qu'il a été défini un peu plus haut dans ce chapitre. On exprime en effet l'action de deux substitutions successives en écrivant l'un après l'autre les symboles qui leur correspondent.
Une transformation est donc un mot dans l'alphabet : 0.

$$0 = \{A, S, G, D, C, R\}.$$

C'est ainsi qu'une transformation couronnée de succès, pour démontrer la formule recherchée, est donnée par le mot

SRGCGDAA.

Par ailleurs, il est clair que certains groupements de substitutions sont identiques. Si l'on désigne l'absence de substitution par le signe « 1 », on a

$$GD = 1$$
$$AS = 1$$
$$C^2 = 1$$

etc.

De plus, si 1 [ ] dénote la longueur de la formule entre crochets il est visible que l'on a

1 [G (formule)] = Z [D (formule)] =1 [C (formule)] =1 [formule]
1 [A (formule)] = Z [formule] -;- t
Z [S (formule)] =1 [R (formule)] =1 [formule] - i.

Toutes ces propriétés peuvent évidemment servir pour éliminer, parmi les mots construits sur l'alphabet O, ceux qui correspondent à des transformations de la formule $(2m + 2n)$ qui ne font pas *avancer* vers la formule $2(m + n)$.
Cela veut dire que même si notre problème peut être transcrit sous la forme d'un problème d'identification de mots, il s'agit d'un problème relatif à une structure de monoïde très *particulière,* concernant l'identification d'un couple très *particulier* de mots. Le théorème de Post-Markov ne s'applique donc pas. Cela n'entraîne pas qu'il n'existe pas de limitations, mais que ces limitations doivent être étudiées et estimées en fonction des données du problème *particulier* devant lequel on se trouve.

Mais de telles limitations, ainsi particularisées, n'ont plus rien de mystérieux et, à première vue, n'introduisent aucune barrière de principe entre l'intelligence naturelle et l'intelligence artificielle. Cependant, trop d'affirmations rapides ont été émises et publiées à ce sujet pour que nous ne nous attardions pas un peu. Nous nous servirons à nouveau d'un exemple.

Il s'agit maintenant d'un jeu assez subtil puisqu'il est lui-même basé sur la notion d'automate, et plus précisément sur la notion de machine de Turing. Une tâche confiée à une telle machine est équivalente à un jeu de cartes de la forme représentée sur la figure 35.

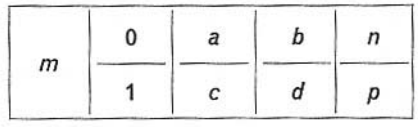


FIG. 35. — Schéma d'une carte dans le jeu de Rado

a, b, c, d sont o ou *1, m, n, p* sont des nombres entiers.

Une telle carte correspond au sous-programme suivant

m : Si la case sous inspection est marquée o, , écrire *a,* effectuer le mouvement correspondant à *b* (avancer si *b = 1,* reculer si b = 0) puis aller à l'instruction numéro *n.*
Si la case sous inspection est *1,* écrire c, effectuer le mouvement correspondant à d, aller à l'instruction numéro p.

Le jeu (inventé par T. Rado) est alors le suivant

1) On choisit $n > o$ et on se donne un ensemble de *n* cartes du type ci-dessus.

2) On constate que la machine de Turing, alimentée par ce jeu de *n* cartes, stoppe après s déplacements.

Le *champion d'ordre n* est celui qui obtient le score a (c'est-à-dire le nombre de *« 1 »* écrit sur la bande le plus élevé), à l'aide de son jeu de cartes M.

Un couple $(M, s)_n$ formé d'un jeu de cartes M et d'un entiers est dit *valide* si M comporte *n* cartes et si la machine s'arrête effectivement après s déplacements.

Le nombre de couples valides pour *n* donné, $N_e(n)$ est toujours fini. On a même

$$Ne(n) < [4(n+1)]^{2n}$$

Considérons alors la fonction

$$\Sigma(n) = \max_{(M, S)_n \text{ valide}} [\sigma]$$

Rado a montré que $E(n)$ (le score maximum que pourrait obtenir un champion d'ordre *n)* est une fonction *non calculable.* En effet, quelle que soit la fonction calculable $f(n)$, on peut démontrer qu'à partir d'un certain *n,* $E(n) > f(n)$.

Ce résultat entraîne le suivant :

Si $S(n)$ est le nombre maximum de déplacements pour une machine appartenant à un couple valide (M, s),,, $S(n) > E(n)$, donc $S(n)$ est également *non calculable.*

Enfin, résultat encore plus remarquable, la fonction $N_e(n)$, définie plus haut, bien que bornée par la fonction relativement simple $[4(n+t)]^{2n}$ est également *non calculable.*

Cet exemple est particulièrement intéressant parce qu'il montre l'impuissance, dans ce cas précis, de la méthode exhaustive » : en effet si on parcourt l'ensemble des nombres entiers de 1 à $[4(n+1)]^{2n}$ on est sûr de rencontrer à un certain instant la valeur de $N_e(n)$. Mais pour savoir quel est cet instant, il faut disposer d'un *raisonnement* dont le nombre d'étapes, lorsque *n* est quelconque, n'a pas de borne supérieure.

LIMITATION DES LIMITATIONS

Peu à peu les menaces que les théorèmes de limitation faisaient peser sur les perspectives de développement d'une intelligence artificielle perdent de leur acuité, mais il semble possible d'aller beaucoup plus loin et de renverser la situation. Auparavant il faut rappeler brièvement la signification de ce que nous venons d'établir

- Tout d'abord, les problèmes de l'intelligence artificielle sont des *problèmes combinatoires.* Une tâche - complètement spécifiée -

peut toujours être accomplie par une recherche exhaustive, c'est-à-dire en examinant successivement tous les éléments d'un ensemble. Mais lorsque la tâche est augmentée, l'ensemble à examiner croît dans des proportions qui font de ce processus une véritable *explosion combinatoire.*

– Les programmes qui *résolvent* les problèmes de l'intelligence artificielle *ne sont pas* des programmes de recherche exhaustive. Au contraire, ils se fondent sur les particularités, les propriétés spécifiques de chaque problème pour développer des techniques de *réduction* de l'expression combinatoire.

- On appelle souvent la méthode exhaustive : « méthode de la force brute ». Il semble normal, par contraste, de caractériser l'intelligence comme *l'aptitude à engendrer ou à utiliser des techniques anti-combinatoires.*

Le lecteur pourra vérifier l'aspect anticombinatoire de la notion d'ouverture dans un jeu comme les échecs, de la propriété de commutativité en algèbre, des règles d'accord en genre et en nombre en linguistique, etc.

Nous allons nous contenter ici d'énumérer quelques cas particuliers qui nous permettront peut-être de rendre plus « parlantes » des notions abstraites comme celles de schéma, de programme, de procédure, etc., dans la mesure où les rapports qu'elles possèdent entre elles seront plus transparents.

« Construire le milieu d'un segment » est un problème particulier dans la classe des problèmes de constructions par la règle et le compas, elle-même famille privilégiée dans l'ensemble des constructions géométriques, etc.

De même, un *schéma de programme* du type

- algorithme :
- documentation de vase :
- données :

devient un *programme* dès qu'on se donne les documents et les données explicitement. Ce programme devient une *procédure si* l'on peut montrer qu'il s'arrête au bout d'un temps déterminé pour produire un résultat. A la limite, le résultat peut être obtenu en introduisant

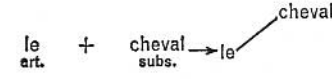
les données dans une simple *formule,* qui devient donc un cas particulier de programme.

Cette remarque permet sans doute d'apporter quelque lumière sur la distinction couramment évoquée - mais mal définie - entre les applications « numériques » et « non numériques».

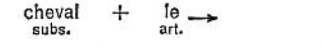
Dans les problèmes numériques, l'algorithme contient de nombreuses formules, arithmétiques ou logiques, et l'on peut tirer parti de l'existence, dans l'automate, de registres liés aux structures arithmétiques et logiques (accumulateurs, registres d'index). Du coup, il devient commode de disposer de langages de programmation qui donnent droit de cité aux formules arithmétiques et logiques dans un format très voisin du format usuel (c'est le cas du Fortran, de l'Algol et de bien d'autres langages).

Dans les problèmes non numériques, les opérations à effectuer sur les données, à l'aide des informations puisées dans la documentation de base, expriment des propriétés du problème, mais des propriétés qui ne sont pas directement liées à celles (commutativité, associativité, etc.) de l'arithmétique et de la logique algébrique.

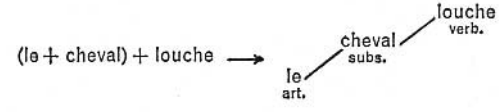
Par exemple, les algèbres que définissent des grammaires de dépendance du type de celles étudiées au chapitre V ne sont en général ni commutatives, ni associatives : on a



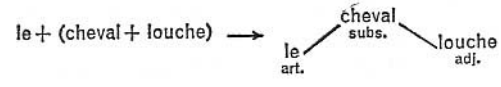
mais



de même



mais :



Or, des *propriétés* comme la commutativité et l'associativité, propriétés qui s'expriment par des formules

$$x + y = y + x$$
$$x + (y + 2) = (x + y) + 2$$

sont, nous venons de le souligner, des programmes d'une nature particulière, des programmes *sous forme concentrée :* elles possèdent un maximum *d'efficacité anticombinatoire.* On peut s'en rendre compte sur l'exemple de G. Teitsin analysé plus haut. Si, aux relations d'équivalence de cet exemple, on ajoute les relations

$$ab \rightleftarrows ba \qquad ae \rightleftarrows ea$$
$$bc \rightleftarrows cb \qquad cd \rightleftarrows dc$$
$$ce \rightleftarrows ec \qquad de \rightleftarrows ed$$

le monoïde est commutatif (on dit aussi « abélien »). Mais maintenant, au lieu, lors des transformations de mots obtenues en utilisant les relations d'équivalences, de se trouver en présence d'un foisonnement incontrôlable de mots nouveaux, la propriété de commutativité permet de réduire tous les mots à un équivalent dans lequel les lettres sont rangées par ordre alphabétique. Dès lors, l'identité des mots se vérifie aisément : le passage du non-commutatif au commutatif a transformé l'indécidable en décidable.

C'est l'absence de symétries, de relations simples dans une structure qui font dire qu'elle est *complexe.* D'après ce qui précède, il est manifeste que la « résolubilité » d'un problème est inversement proportionnelle à sa complexité. Par contre, la capacité « intellectuelle » d'un automate va de pair avec sa propre complexité.

S'attaquer au bilan « problème-automate » du point de vue de la complexité, c'est donc finalement tenter d'établir, sur une base scientifique, le diagnostic des possibilités de l'intelligence artificielle.

Pour que le bilan puisse être précis, il faut lui fournir une base quantitative, d'où l'importance que l'on doit attacher à une mesure quantitative de la complexité (d'un problème ou d'un automate). Ici, nous atteignons les frontières de la science actuelle. Il existe de nombreux travaux en cours dans ce domaine (principalement en U.R.S.S.). Mais avant de les examiner, nous allons revenir une dernière fois sur les « théorèmes de limitation » car nous pensons que ces théorèmes,

loin d'apporter des constats d'échecs, sont un point de départ indispensable pour le développement d'une conception positive de la complexité.

* * *

On sait que l'apparition de paradoxes dans la théorie « naïve » des ensembles est à l'origine de *l'effort* de formalisation de la mathématique et de la logique modernes. Il est moins connu, mais au moins aussi important, que les mêmes paradoxes sont en réalité la racine des théorèmes de limitations. C'est ainsi que le fameux paradoxe de l'Épiménide (ou du « menteur ») : « Épiménide le Crétois dit que tous les Crétois sont menteurs», énoncé dont il est évidemment impossible d'apprécier la vérité ou la fausseté, sert de modèle à la démonstration du théorème de Gödel. Le paradoxe de Richard (que nous n'analyserons pas ici) sert de schéma aux démonstrations des théorèmes de Church, Kleene et Turing.

Pour comprendre la signification de ces paradoxes, à l'aide des images et des concepts auxquels nous nous sommes habitués, il est commode d'imaginer le problème : «quelle est la valeur logique de l'affirmation de l'Epiménide ? » comme une succession de réponses contradictoires, chaque conclusion inversant la prémisse sur laquelle elle se fonde. Cela évoque un programme pour automate qui contiendrait la suite d'instruction

n Exécuter l'instruction n° $n + 1$
$n + 1$ Exécuter l'instruction n° n

Un tel programme ferait évidemment « cycler » l'automate indéfiniment (1).

Considérons maintenant le paradoxe de Berry qui s'énonce comme suit : « le nombre de Berry est le plus petit nombre naturel non définissable au moyen d'une phrase contenant 50 mots au maximum pris dans un vocabulaire ». Comme la phrase précédente ne comprend que 37 mots le nombre de Berry est contradictoire.

(1) On trouverait des exemples détaillés de ce genre dans l'article de S. Gorn « The treatment of ambiguity and paradox in mechanical languages » dans *Proceeding of Symp. in pure Math., V,* 1962, p. 201.

Bien entendu la clef du paradoxe réside dans la constatation de ce que la notion de « définissabilité » utilisée dans la phrase en question n'est pas elle-même clairement définie. On utilise en effet le langage ordinaire pour construire un objet mathématique à qui l'on voudrait ensuite imposer des contraintes de rigueur qui ne sont applicables qu'aux définitions formelles.

Le paradoxe disparaît, en effet, dès qu'on se donne un système formel bien défini A qui contienne les symboles de l'arithmétique. Car dès lors le nombre de Berry correspondant au système A (c'est-à-dire le plus petit nombre naturel non définissable au moyen d'une *expression du système A* contenant 50 symboles élémentaires au maximum) n'est pas contradictoire.

Mais on peut aller plus loin et se servir de l'idée de Berry pour construire une notion mathématique nouvelle : c'est ce que nous proposons d'appeler la fonction de « Berry-Beth » (1) : « La fonction de Berry-Beth qui correspond au système formel A est la fonction f qui associe à un nombre naturel m, le plus petit nombre naturel n non définissable au moyen d'une expression du système A contenant m symboles élémentaires au maximum. » On peut montrer que la fonction f *ne peut pas* être définie formellement à l'aide du seul outillage symbolique contenu dans le système A. Pour définit f de façon complète, il faudrait disposer d'un système formel B plus puissant que A, etc. L'impossibilité de définir f dans A est un résultat *négatif,* caractéristique des résultats négatifs que sont les théorèmes de limitation. Mais on voit apparaître ici la possibilité de *renverser* le sens des applications de ce genre de théorèmes.

En effet les autres faits de limitation comme le théorème de Gödel mettent en oeuvre des énoncés mathématiques «monstrueux », expressions qui expriment leur propre non-déductibilité, etc. Bien que monstrueux ces énoncés appartiennent à un corps d'énoncés licites et les conséquences, négatives, de leur mise en oeuvre n'en sont pas moins suffisantes pour fonder les limitations. Par contre les êtres mathématiques comme la fonction de Berry-Beth se situent dans

(1) Cf. E. W. BETH, Les fondements logiques des mathématiques, Gauthier-Villars, 1955, p. 194

une une classe beaucoup plus attirante : celle des mesures de construction d'autres êtres (normaux, ceux-là) de la mathématique usuelle.

On peut alors s'attendre, dans cette direction, à l'apparition de résultats *positifs,* c'est-à-dire qui expriment une *limitation des faits de limitation : s'il* existe bien des problèmes trop complexes pour un système combinatoire donné, il existe *toujours* une réduction de la complexité permettant de résoudre le problème (démontrer - ou réfuter - un théorème), en augmentant la puissance du système, ou au contraire en diminuant le degré de généralité du problème posé, par exemple en remplaçant le problème général des mots par celui d'une classe particulière de mots.

LES THÉORIES DE LA COMPLEXITÉ

La notion générale de complexité est évidemment fort ancienne. Mais elle est restée, jusqu'à une période très récente, une notion qualitative, permettant d'exprimer un sentiment de difficulté, voire d'impuissance.

Mais à partir du moment où l'on s'est aperçu que beaucoup de notions qui appartenaient au langage quotidien étaient susceptibles d'une analyse logique, et, moyennant quelques conventions restrictives, pouvaient s'intégrer dans un système formel, il était naturel d'examiner l'un après l'autre les différents concepts qui se situaient dans le « voisinage sémantique » des concepts déjà formalisés.

C'est ainsi que la logique des propositions ayant exploité des conjonctions comme « et » et « ou », le calcul des prédicats ayant intégré des expressions telles que « quelle que soit» etc., le formalisme s'attaqua à des concepts plus larges comme celui de « déductibilité » et de « démonstrabilité ».

Il restait cependant un grand pas à franchir entre la formalisation de tels concepts susceptibles d'aboutir à des prédicats qui ne peuvent avoir que les valeurs « oui » ou « non » et celui de la complexité qui, s'il était convenablement formalisé devrait déboucher sur une échelle *quantitative* de valuations.

Le premier pas dans cette direction a été - du moins à notre connaissance - accompli par le polonais A. Lindenbaum dans une commu-

nication au Congrès international de Philosophie tenu à Paris en 1935 (1).

La disparition de Lindenbaum au cours de la seconde guerre mondiale l'empêche de développer son projet qui fut repris indépendamment par N. Goodman, à partir de 1943 (2).

Pourtant Goodman ne put mener très loin son projet (limité à la complexité de ce qu'il appelait les « bases extra-logiques », mais qui correspond, dans le langage Bourbakiste, aux espèces de structures algébriques).

Malgré cet échec, la notion de complexité était étudiée implicitement par les logiciens qui, après Gödel étudiaient les «hiérarchies » constructives (fonctions récursives primitives, générales, arithmétiques, analytiques, etc.). D'ailleurs l'idée d'utiliser la notion de « longueur d'une démonstration » était présente chez Gödel dès 1930.

Mais c'est tout récemment que l'étude de la complexité a pris un nouvel essor, notamment en U.R.S.S. et ceci sur deux plans relativement indépendants : étude de la complexité des réseaux de relais avec Iablonski, Lupanov et bien d'autres ; étude des sous-classes de machine de Turing définies par des contraintes particulières sur la longueur de bande utilisée, ou le temps de fonctionnement autorisé (travaux de Ritchie, Yamada, Schutzenberger, Trahtenbrot, Rabin, etc.) (3).

Après ces développements - dont nous souhaitons qu'ils n'aient pas, à leur tour, semblé trop complexes ! - nous croyons être en mesure de répondre au problème posé dans ce chapitre, justifiant ainsi l'ouvrage tout entier.

Car dès le chapitre II, et à chaque fois qu'un exemple nouveau était présenté, nous avons tenu à faire remarquer l'identité foncière de tous les problèmes posés. Tous peuvent, en fin de compte, être présentés comme des problèmes d'identification de mots dans un certain alphabet, en conjonction avec l'existence de relation de substitution, ou encore la recherche d'un chemin dans un graphe.

(1) *Cf. Actualités scientifiques et industrielles, n° 394, Hermann, 1936, p. 29.*

(2) On peut trouver une analyse succincte des travaux de Goodman et de ses continuateurs (Kemeny, Svenonius) dans un article de R. RUDNER *paru dans Philosophy of Science, 28, 1961, p. 109.*

(3) Sur tout ceci voir notre exposé au Congrès d'Amsterdam, août 1967.

Suivant l'exemple choisi, le système de relations était énoncé plus ou moins complètement.

De même, le mot à comparer pouvait être donné explicitement ou seulement par un certain nombre de ses propriétés (de même pour le chemin). On se trouvait donc bien plus souvent en face d'une *classe* de problèmes que d'un problème *unique.*

Cette situation a un équivalent, bien entendu, dans le langage des automates et de leurs programmes, et nous avons décrit la situation avec quelques détails à la fin du chapitre II, en présentant - pour les distinguer - les notions de tâche, de programme, de moniteur et de système.

Lorsqu'une tâche est complètement définie, on peut écrire le programme et introduire les données initiales. Le seul obstacle qui peut se dresser est alors un obstacle matériel (encombrement possible de la mémoire, etc.), comme dans l'exemple a) ci-dessus. Lorsque la tâche n'est pas complètement définie, que le problème posé est en fait, non *d'accomplir une tâche,* mais de trouver une *méthode pour accomplir toute une classe de tâches, il* n'est pas possible, bien souvent, de spécifier les contraintes de temps ou de mémoire qui vont peser sur l'automate.

Dans certains cas heureux, il est possible de caractériser chaque tâche de la famille par un paramètre numérique *n.* La « taille » de l'automate sera alors une fonction de ce paramètre numérique. Il sera peut-être alors possible d'examiner comment la taille de l'automate varie (c'est-à-dire, en général, grandit) en fonction de *n. Si* la fonction croît démesurément avec n, on peut être assuré qu'à partir d'une certaine valeur, il n'y aura plus d'automate suffisant à accomplir la tâche attendue.

Le problème, qui est donc une classe de «tâches», est alors *indécidable.*

Mais s'il est extrêmement agréable de pouvoir disposer d'une méthode générale pour résoudre une classe immense de tâches, on peut bien accepter la solution moins élégante, mais bonne à prendre s'il n'y en a pas d'autres, qui consiste à diviser la classe en sous-classes : et même en autant de sous-classes qu'il est nécessaire pour que chacune d'entre elles puisse être pourvue d'une méthode générale de solution.

A ce point de l'exposé le lecteur aura probablement formé de lui-même l'idée que la clé du problème de réalisabilité en intelligence artificielle

réside dans la mise en correspondance de deux hiérarchies : celle des problèmes et celle des solutions.
Nous formulerons cette thèse sous la forme plus explicite que voici

A tout problème posé dans un système formel complètement défini, on peut associer un automate et un programme pour cet automate. L'automate atteindra effectivement la solution s'il dispose d'une structure dont la complexité soit au moins égale à celle du problème posé. S'il n'en est pas ainsi, il est cependant toujours possible de revenir à des conditions de résolubilité

- soit en augmentant la complexité de l'automate ;
- soit en diminuant le degré de généralité du problème.

Le lecteur pourra vérifier la validité de cet énoncé (que nous avons laissé un peu vague, puisqu'en tout cas, nous n'en proposons pas ici une démonstration), sur un exemple tel que celui de l'analyse grammaticale (on augmente la complexité de l'automate en augmentant le nombre des règles qui constituent sa « documentation de base », on diminue la généralité du problème en imposant une borne supérieure à la longueur des syntagmes).
Finalement, nous pensons que la notion de complexité donne la clé d'une autre notion que nous avons évoquée plusieurs fois, notion d'ailleurs fort controversée, qui est celle *d'heuristique.*
Par définition, la méthode heuristique est celle qui n'est pas fondée sur la mise en oeuvre mécanique d'une jeu de formules, mais sur l'exploration guidée des possibilités du problème. Comment guider cette exploration (car il ne s'agit évidemment pas de l'exploration exhaustive), autrement que par une intuition, une analogie, etc. ? L'heuristique n'est-elle pas, par conséquent, typique des capacités de l'homme, par contraste avec celle des automates ?
Et, de fait, les méthodes dites heuristiques introduites dans certaines recherches relevant de l'intelligence artificielle, n'ont pas entraîné la conviction (il s'agit notamment des points de vue développés par Newell et Simon pour le jeu d'échecs, et par Gelernter pour la démonstration des théorèmes de géométrie). Les critiques ont pu montrer que le guidage non formalisé dans la recherche des solutions était, soit inopérant soit informé par avance des solutions à obtenir.
Il nous semble que les remarques présentées au cours de ce chapitre

permettent de présenter une heuristique automatique parfaitement fondée.
Pour le voir, il est commode de représenter la résolution du problème comme la recherche d'un chemin dans un graphe. Si l'on disposait d'un jeu de formules donnant la solution, le graphe serait linéaire ou au moins aurait l'allure d'un treillis.
Dans la méthode exhaustive, tous les chemins, sans exception sont explorés jusqu'à ce qu'un premier chemin victorieux soit découvert.
Une méthode heuristique élimine, à chaque carrefour, un nombre suffisant de chemins possibles pour que l'exploration ne conduise pas à une explosion combinatoire du nombre d'opérations à effectuer.
Ce filtrage sera aisé si l'on dispose d'une mesure adéquate de la complexité : on éliminera les chemins qui accroissent la complexité du chemin restant.
On vérifiera que l'application de cette procédure redonne bien les limitations relatives que l'on avait rencontrées un peu plus haut.

CHAPITRE VII

La tête au-delà des murs

Partis de la constatation empirique qu'une communauté de problèmes et de méthodes s'est constituée et est couramment reconnue sous le nom d' « intelligence artificielle », nous avons entrepris, au cours de cet ouvrage, de donner à cette communauté un fondement explicite, et d'en tirer les conséquences.

Pour cela, au cours du chapitre II, nous avons présenté les éléments d'une formalisation - langage des monoïdes et représentation à l'aide de graphes - à l'aide de laquelle les divers problèmes que nous avons rencontrés s'expriment très naturellement.

Puis, au cours des chapitres III, IV, et V, nous avons exploré les principales rubriques de l'intelligence artificielle en développant quelques exemples et en donnant une idée des résultats obtenus. A cette occasion on a pu constater la similitude des techniques utilisées pour la recherche des solutions, la construction des programmes, etc.

Enfin, dans le chapitre précédent, nous avons complété cet effort de synthèse en explicitant une notion - celle de complexité - qui est le dénominateur commun à toutes les branches et sous-branches de la communauté des problèmes que nous avions parcourue et qui, en même temps, fournit la base d'une estimation qualitative et parfois même quantitative des performances que l'on doit attendre des automates pour que les problèmes soient non plus seulement *posés,* mais aussi *résolus.*

En s'appuyant d'une part sur ces estimations, et d'autre part sur les résultats expérimentaux obtenus ici et là, et auxquels nous avons fait allusion chemin faisant, il est possible d'établir un bilan et de dresser des perspectives. Et c'est ce que nous allons tenter de faire au cours du présent chapitre.

On peut le dire en quelques mots : le bilan actuel des réalisations *pratiques* de l'intelligence artificielle est modeste.

Certes, dans chacun des domaines que nous avons abordés de telles réalisations existent. Dans plusieurs institutions il existe des programmes avec qui on peut jouer aux échecs (le plus connu est celui du Massachusetts Instituts of Technology) (1) ; nous avons nous-même écrit un programme qui joue au Go-Bang. Il existe une demi-douzaine de programmes de démonstration automatique des théorèmes (il faut citer notamment ceux développés à Argonne par J. Robinson, et à M.I.T.R.E. par J. Friedmann). Plusieurs programmes de traduction automatique du russe vers l'anglais sont en exploitation (en particulier au Centre de Recherches d'Euratom à Ispra on utilise une version retouchée et complétée du célèbre programme mis au point à l'Université Georgetown de Washington, par une équipe - aujourd'hui dispersée - dirigée par G. Dostert).

Mais il est clair que l'utilisation des calculatrices électroniques pour des tâches de cet ordre - en comparaison avec leur utilisation pour des tâches de calcul numérique - reste négligeable.

De même les équipes engagées dans des recherches de ce type sont encore peu nombreuses. Les crédits accordés sont souvent importants mais la politique des bailleurs de fonds (en particulier des diverses administrations civiles et militaires aux États-Unis) demeure hésitante, ce qui compromet le recrutement et la formation des chercheurs.

Et pourtant nous avons intitulé ce chapitre «la tête au-delà des murs» pour indiquer avec suffisamment de force les perspectives qui - croyons-nous - sont dès à présent discernables.

Nous allons tout d'abord analyser les conditions générales qui règnent aujourd'hui sur le marché des machines à calculer électroniques,

(1) Repris et développé à l'Université de Stanford.

conditions qui - indépendamment des fluctuations de la mode - exercent une pression constante en faveur de développements nouveaux et de progrès de toutes sortes. Ces conditions sont celles d'un engouement généralisé pour l'automatisation et les seules contraintes qui le limitent sont celles relatives à la formation du personnel spécialisé.

Puis nous examinerons quelques domaines d'activité qui ne relèvent pas directement de l'intelligence artificielle, mais qui, dans la mesure où des améliorations de plus en plus importantes sont considérées comme souhaitables, posent des problèmes qui rejoignent peu à peu notre domaine. Il s'agit des problèmes de la gestion automatisée, du contrôle industriel et de ce qu'on appelle le *software,* nom sophistiqué pour « l'art de la programmation».

Nous aborderons alors un domaine qui se trouve à cheval sur les domaines de l'exploitation et de la recherche. Il s'agit du domaine de la linguistique appliquée dont une partie dépend de développements du type de ceux que nous avons présentés au cours du chapitre V, mais dont une autre se situe à un niveau plus élémentaire. L'interdépendance de ces deux types de problèmes est évidemment un intéressant facteur de progrès.

Enfin nous examinerons brièvement le domaine de ce qu'on pourrait appeler paradoxalement « l'art artificiel », domaine qui n'est pour le moment qu'une curiosité, mais qui pourrait bien donner naissance à d'importants développements.

LA RUÉE VERS LES MACHINES

La discussion que nous avons conduite au chapitre précédent nous a permis de conclure qu'aucune raison de principe, et en particulier aucun théorème de logique mathématique, ne s'opposait au développement continu de l'intelligence artificielle. Certes, les progrès technologiques ne peuvent améliorer indéfiniment les performances des machines. La nanoseconde (10^{-9} seconde) semble être un temps minimum pour une opération élémentaire qu'il sera impossible de dépasser et peut-être même d'atteindre (les performances actuelles atteignent 100 nanosecondes, c'est-à-dire 10^{-7} seconde). Par ailleurs

il n'est pas raisonnable de compter sur un fonctionnement continu d'une calculatrice sur un même problème pendant des périodes d'un ordre de grandeur supérieur à la journée.

Cela veut dire qu'on ne peut guère espérer résoudre - même dans un avenir éloigné - des problèmes qui nécessiteraient l'exécution d'un nombre d'opérations élémentaires dépassant 10^{14} et il est évident qu'on peut imaginer de tels problèmes (par exemple calculer les 10^{15} premiers nombres entiers et les imprimer en représentation décimale). Mais des problèmes de ce genre sont des problèmes combinatoires.

Nous pensons avoir montré que ces problèmes ne relèvent de l'intelligence artificielle que dans la mesure où il est possible d'y définir des procédures anticombinatoires. De telles procédures réclament bien souvent l'utilisation d'un stock d'information important et complexe (par exemple un réseau sémantique sera nécessaire pour la constitution d'un programme de traduction automatique rigoureux et efficace).

Cela veut dire que le travail important et difficile n'est pas celui de la calculatrice, mais celui des experts chargés de découvrir les procédures et de rassembler les matériaux constituant le stock d'informations, de l'organiser pour son exploitation optimale, etc. De tels travaux réclament de la patience, de l'ingéniosité et, par-dessus tout, une très grande quantité de travail. Il n'y a pas de voie royale pour l'intelligence artificielle.

Ce que nous venons de dire implique qu'un progrès vraiment spectaculaire ne sera accompli que lorsque des moyens suffisants, en particulier des équipes fortes et animées d'un esprit de collaboration (esprit qui, jusqu'à présent a bien souvent manqué, notamment dans le domaine de la traduction automatique) seront rassemblés.

Or, nous sommes convaincus que, après une période de difficultés diverses, ces moyens seront effectivement rassemblés et les résultats spectaculaires qu'on doit en attendre, atteints. Pourquoi cette conviction ?

C'est qu'il existe aujourd'hui, *en deça* du domaine de l'intelligence artificielle, un domaine d'utilisation des calculatrices électroniques en pleine expansion, domaine dont le développement exigera le recours aux procédés et aux résultats de l'intelligence artificielle. Il en résul-

tera tôt ou tard une pression de la demande qui, plus que les initiatives individuelles, permettra d'obtenir les investissements nécessaires au progrès.

L'un des caractères notables de notre époque, c'est en effet l'existence d'une véritable *explosion informationnelle. Il* est significatif, à cet égard, qu'au moment du récent congrès de l'I.F.I.P. (Fédération internationale pour le traitement de l'information), tenu à New York en mai 1965, le *New York Times* ait publié un supplément intitulé *The Information Revolution.* Nous sommes à l'ère des institutions, des traités, des règlements, des systèmes comptables, du contrôle formalisé. Nous sommes aussi à l'ère des mesures numériques, de l'estimation statistique, de la prévision et de la planification. Nous sommes enfin à l'ère du contrôle automatique.

A ces multiples activités correspondent des circuits informationnels multiples : érection et codification d'informations (fichiers individuels, enregistrements de résultats de mesure, etc.), circulation d'information (sous forme manuscrite, imprimée, téléphonique, carte perforée, bande magnétique, etc.). Les calculatrices électroniques ne sont pas seulement - nous l'avons vu - des machines à calculer. Elles sont aussi des machines à traiter l'information. Aussi était-il naturel de les utiliser pour l'automatisation des travaux de codification et de circulation que nous venons d'évoquer. Il est alors apparu rapidement que la clientèle de ces machines serait principalement une clientèle intéressée par des applications de cette sorte.

Nous allons illustrer tout cela en examinant un peu plus en détail ce domaine que l'on pourrait qualifier de «préliminaire à l'intelligence artificielle ».

LA GESTION AUTOMATISÉE

La première conquête des machines est celle de la gestion administrative et commerciale. Il s'agit de l'enregistrement, de la manipulation et de l'évaluation d'objets de divers types

- des biens (magasin, inventaire, achats, etc.) ;
- des personnes (établissement des bulletins de paie, dossiers médicaux, etc.) ;

- des actes comportant l'intervention simultanée de biens et de personnes (gestion des contrats, des programmes).

A ce niveau élémentaire (tout au moins du point de vue de la structure) il s'agit d'assurer des codifications efficaces tant pour l'enregistrement des données que pour la formulation des problèmes posés par les administrateurs, comptables, agents de contrats, etc. L'aspect combinatoire découle simplement de la masse considérable de données à manipuler. En un mot, les fichiers sont très épais, les programmes de traitement relativement minces.

A un niveau supérieur, les mêmes données sont utilisées pour la constitution de dossiers statistiques. Ces statistiques doivent alimenter à la fois l'analyse de l'évolution déjà accomplie (recherche de tendances, analyse de régression, analyse factorielle, etc.) et la prévision.

C'est dans cet esprit et avec ces techniques que se développent les systèmes permettant, sinon la gestion automatique des grands projets, tout au moins le dialogue du chef de projet avec la calculatrice. Les techniques connues sous le nom de « recherche de chemin critique », « programme Pert », etc., appartiennent à cette catégorie.

Ici encore les données à traiter sont très nombreuses, mais en plus la manipulation de ces données fait intervenir des procédures (notamment mathématiques et plus précisément statistiques) beaucoup plus évoluées que celles du niveau précédent.

Enfin, au niveau le plus élevé - dans ce domaine des applications économiques et de gestion - se situe la simulation : simulation d'entreprise ou d'administration, simulation de réseau d'entreprises, simulation d'économie régionale, nationale ou même multinationale. A ce niveau, on voit s'ajouter à la masse des données et à la précision mathématique des techniques d'estimation, l'agencement logique et chronologique des actions et échanges dont la mise en ouvre articulée est caractéristique de l'ensemble que l'on simule.

Au niveau de la gestion du personnel et du budget les difficultés du travail de l'automate sont essentiellement liées à la nécessité d'une

multitude de vérifications et en particulier vérification des données. Les calculs sont en effet trop élémentaires pour qu'une erreur ait une chance sérieuse d'y apparaître. Cependant ce travail de vérification est suffisamment fastidieux pour qu'on tente de le rendre moins pénible par l'utilisation de langages de programmation spécialisés. Il en va de même pour les problèmes du type « recherche opérationnelle ».

Dès qu'on aborde la simulation des grands ensembles industriels, les jeux stratégiques, etc., le problème de la vérification des données cesse d'être le plus important. Car les *actions* à simuler sont soumises au contrôle d'une certaine logique, logique dont les diverses articulations doivent être reproduites fidèlement par l'automate.

Par exemple la simulation d'un supermarché exige la décomposition de son activité en actions élémentaires : arrivée du client, achats, passage à la caisse, départ, etc. La logique exige que le client ne passe à la caisse qu'après ses achats et non avant, qu'il parte après son passage à la caisse et non avant. D'autre part, il existe moins de caisses que de clients, d'où l'existence de queues, d'où la nécessité d'attribuer un certain temps aux opérations élémentaires, etc.

On voit qu'ici la difficulté dans le travail de l'automate prend un aspect beaucoup plus logique que dans les exemples précédents.

L'aspect « intelligence artificielle », c'est donc ici la constitution de systèmes de programmation spécialisés dont la structure exprime un peu des propriétés générales des systèmes à simuler. On citera, à titre d'exemple, le G.P.S.S. (General Problem Storing System) mis au point à I.B.M. par Gordon. Dans un tel système, il sera aisé de simuler le fonctionnement d'un supermarché tel que celui décrit par le diagramme ci-contre (fig. 36).

On dispose, en effet, dans le G.P.S.S., de sous-programmes standards qui simulent l'arrivée au hasard d'individus, la réalisation (qui dure un certain temps) de transactions élémentaires, le déplacement d'individus ou de marchandises, etc., le tout lié par un système de contraintes qui est caractéristique de l'établissement dans lequel ces transactions ou ces déplacements ont lieu.

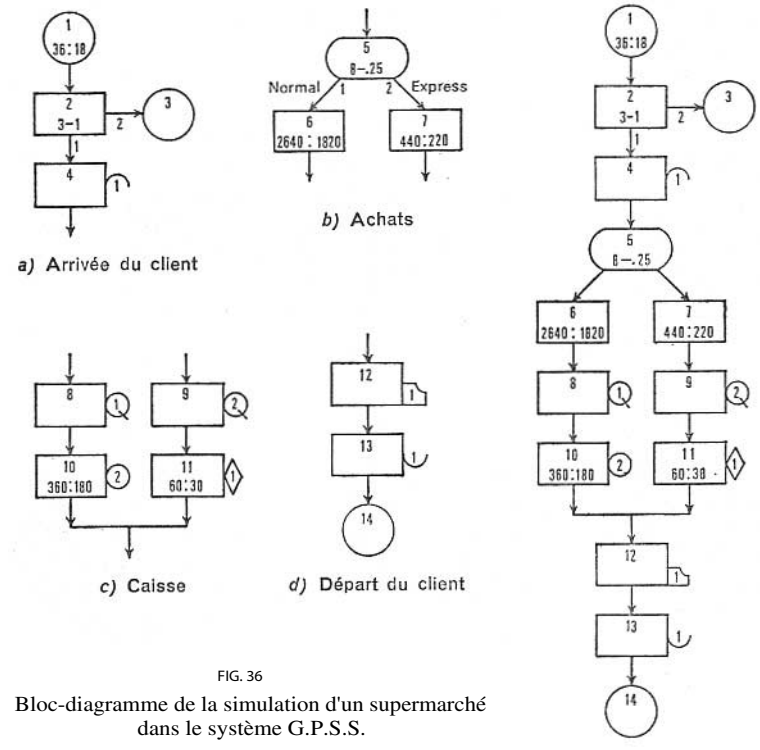


FIG. 36

Bloc-diagramme de la simulation d'un supermarché dans le système G.P.S.S.

LE CONTROLE INDUSTRIEL

La seconde conquête des calculatrices est celle du domaine industriel et technologique. Cette conquête est plus récente et, notamment en Europe, n'est pas achevée. Elle intéresse, elle aussi, plusieurs niveaux que nous aborderons également dans l'ordre de la complexité croissante.

Au premier niveau, on voit apparaître, dans les boucles de contrôle automatique des ensembles industriels, des appareils qui transforment

les informations, effectuent des calculs élémentaires, des comparaisons, etc., et émettent des signaux destinés à la commande d'organes divers disposés sur la bande. Ce sont des calculatrices spécialisées et de taille souvent modeste.

Au fond le régulateur de Watt en est l'ancêtre : on sait que c'est le régulateur de vitesse des machines à vapeur qui tourne à la vitesse du moteur et où la force centrifuge permet l'ouverture d'une vanne qui contrôle le débit de la vapeur. Ici pas de calcul explicite, mais le régulateur assure la résolution d'une équation implicite.

Au deuxième niveau, la calculatrice de contrôle prend une envergure nouvelle, elle dirige les divers organes non plus en fonction de consignes locales qui ont un caractère « réflexe », mais en fonction d'une consigne *d'optimisation* dont la mise en oeuvre nécessite à la fois l'accumulation d'un grand nombre d'informations, mais aussi la résolution de problèmes analytiques, numériques et logiques fort avancés.

Les calculatrices qui correspondent à ce niveau sont spécialisées en ce sens qu'elles disposent en général d'organes d'entrée et de sortie particuliers (par exemple des transducteurs permettant la transformation de signaux analogiques en signaux digitaux et réciproquement). Elles sont également étudiées pour fonctionner en « temps réel ». D'autre part elles sont universelles en ce sens qu'une même calculatrice de contrôle peut être utilisée dans le voisinage d'un réacteur nucléaire, d'une installation de cracking, d'un laminoir, etc.

Lorsqu'on aborde le troisième niveau (ce qui n'a lieu qu'en quelques points d'avant-garde de l'industrie américaine), on voit le domaine de la recherche et du développement s'ajouter à celui de la production. C'est ainsi que les calculatrices modernes sont étudiées, simulées puis dessinées à l'aide de calculatrices.

Des organes périphériques spécialisés se chargent même d'effectuer les dessins cotés, de les photographier et d'en tirer les « bleus » qui sont livrés aux ateliers ou, mieux, les bandes perforées qui commanderont les machines-outils.

Ici encore, on arrive bien près du domaine de l'intelligence artificielle. En effet, la machine doit traiter des informations à la fois nombreuses et très structurées. D'ailleurs, on voit surgir, ici également, des langages de programmation spécialisés. Nous citerons un exemple le Lotis (un langage formel pour décrire la LOgique, le « TIming »

et le « Sequencing » des machines). Ce langage, mis au point par H. P. Schlaeffi, permet à la fois de décrire formellement un automate (mémoire, structure des registres, organisation spatio-temporelle, etc.) et d'en simuler le comportement sur une autre machine.

A titre d'illustration, nous présentons ci-dessous la description formelle d'une calculatrice dont la mémoire posséderait 10^{14} mots de 32 bits chacun, avec un cycle et un temps d'accès de 12 et 5 « unités de temps », etc.

```
cpu abc /
m(14b, 32); mx(7, 14); ar(14); sr(14);
ir(32) = [iop(8), tag(1), x(3), nil(6),
   iad(14)] ;
ic(14); ac(32); md(32); ov(1); ready(1);
+ (4) ; ∨(1) ; ≠ (1) ;

fct read, core /
1, 5 : ready : = 0 ; sr : = m(ar) /
2, 7 : ready : = 1 / fin
fct store, core /
1, 5 : ready : = 0 /
2, 7 : ready : = 1 ; m(ar) : = sr / fin
```

```
seq instrfetch, ctr /
1, ready : ar : = ic ; ic : = ic + 1 /
2, core : call read /
3, ready : ir : = sr /
4 : ar : = iad + (if∨x then mx(x)
     else 0) ;
   if not tag then goto 7 /
5, core : call read /
6, ready : ir[8, 31] : = sr[8, 31]; goto 4 /
7 : if iop[0] then (if arit then call
                 arit : = iop[1, 7] else
                 hold)
            else goto iop[1, 7] / fin

seq add, arit /
1, core : call read /
2, ready : md : = sr /
3, 8 : ov : = (ac[0] ≠ md[0]); ac : =
   ac + md /
4 : if not ov then ov : = (ac[0] ≠
   md[0]) else ov : = 0 ;
   call instrfetch, 1 / fin.
```

Nous ne donnerons pas ici la signification des différents symboles utilisés car il s'agit plutôt de donner une idée de ce que sont les recherches de ce type.

LE NOUVEAU « SOFTWARE »

La troisième conquête des machines commence à peine et, en particulier, elle n'a guère pénétré notre continent, mais on peut prévoir un déferlement considérable dans les années qui viennent. Il s'agit

de la conquête des professions libérales ou intellectuelles. Nous préciserons à l'aide d'exemples ce que nous entendons par là et, une fois de plus, en distinguant des niveaux de difficulté croissante. Nous nous précisons au départ qu'il ne s'agit pas ici de *remplacer* le travailleur intellectuel, mais de l'aider. Nous sommes toujours dans la *banlieue* de l'intelligence artificielle.

Au niveau le plus élémentaire (qui est, bien entendu, le plus développé), il faut bien citer la programmation automatique. Nous avons indiqué à plusieurs reprises l'apparition de langages spécialisés pour différents domaines d'application des machines. Il est clair que cette aide au programmeur n'est obtenue que grâce à la mise en oeuvre de programmes spéciaux (les compilateurs) qui effectuent la traduction langage spécialisé-langage machine. Ces programmes de manipulation de symboles comportent souvent des dizaines de milliers d'instructions. Leur étude systématique, la recherche de programmes capables, si on leur donne la description formelle d'un langage et d'une machine, de produire le compilateur qui permettra à la machine d'interpréter le langage, tout cela est donc un *effort* du même ordre que ceux que l'on doit fournir dans les diverses disciplines qui relèvent de l'intelligence artificielle.

On donne à ce domaine et à l'objet de ses recherches le nom de *software,* néologisme anglais basé sur un jeu de mots. En effet, les machines à calculer elles-mêmes, c'est-à-dire l'outillage électronique et mécanique qui les compose, sont désignées en anglais par le mot *hardware* qui veut dire « quincaillerie ». Mais l'adjectif *hard* qui figure dans *hardware* est le symétrique de *soft* dans la mesure où des adjectifs signifient respectivement « dur » et « doux » (1). Le *software* c'est donc ce qui, dans le fonctionnement des calculatrices, est extérieur à l'appareillage lui-même : c'est l'ensemble des problèmes liés à la programmation.

Le premier effort dans la constitution du *software* a été la mise au point de langages de programmation plus voisins du langage des mathématiciens que ne l'était le langage des machines. Nous ne nous étendrons pas sur ce point qui a été abondamment traité par des

(1) Et pourquoi ne pas substituer au couple « hardware-software , un couple plus attrayant tel que « plumage-ramage » ?

auteurs excellents (1). Rappelons simplement que les premiers langages de programmation étaient destinés à faciliter l'écriture de programmes comportant un grand nombre de formules algébriques Fortran, Mad, Algol sont de ce type. Puis on s'est intéressé à l'écriture de programmes comportant des instructions du type comptable, ce qui a donné naissance au Cobol.

Parallèlement on a créé des langages spécialisés relatifs à des classes plus restreintes de problèmes. Nous en avons indiqué ci-dessus quelques exemples : G. P. S. S., Lotis, etc.

Enfin on a développé des langages aptes au traitement des *listes.* En programmation une liste n'est pas une suite d'informations, mais une organisation comme celle de la figure 37.

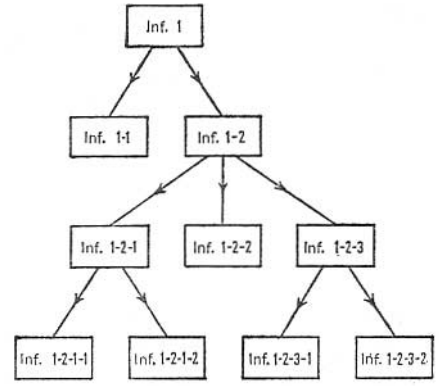


FIG. 37. — Organisation d'informations sous forme de « liste »

Des langages spéciaux ont été créés qui permettent d'accomplir aisément des opérations telles que :

- introduire de nouvelles informations dans la liste ;
- effacer certaines informations dans la liste ;
- chercher dans la liste une information donnée, etc.

Ces langages sont le F.L.P.L., les I.P.L., le L.I.S.P., etc.

I.B.M. a récemment développé un langage nommé P.L. 1 qui s'efforce de combiner les avantages du Fortran, de l'Algol et des langages de traitement de listes.

Pour chaque type de machine, l'utilisation d'un de ces langages nécessite l'utilisation d'un programme particulier, appelé *compilateur* qui traduit les expressions écrites dans le langage en instructions accep-

(1) Voir en particulier le livre de M. et Mme POYEN cité en bibliographie.

tées par la machine. La multiplication des langages et des machines rend ce travail gigantesque au point qu'il essouffle même les constructeurs les plus puissants. On en est donc arrivé au point où il serait souhaitable d'avoir des programmes qui construisent automatiquement les compilateurs quand on leur donne la structure de la machine et celle du langage.

De telles recherches sont en cours activement. Elles mettent en jeu un certain nombre de domaines (logique, mathématique, analyse combinatoire, etc.) qui, exploités avec l'esprit de recherche d'encombrement minimum qu'on imagine sans peine, rejoignent évidemment le domaine de l'intelligence artificielle.

LA LINGUISTIQUE APPLIQUÉE ET LES DOMAINES CONNEXES

Au niveau suivant, la manipulation de programmes est remplacée par la manipulation de textes.

Conformément aux remarques que nous avons présentées à la fin du chapitre V, on ne peut pas s'attendre, dans l'état actuel des choses, à ce que les textes soient exploités dans la totalité de leur contenu. On doit se borner à traiter certains aspects du langage (ceux qui sont le plus aisément formalisables) ou à traiter des *parties* du langage, ou enfin à utiliser les programmes comme auxiliaires du travailleur humain. Mais dans les trois cas il existe des applications suffisamment intéressantes et importantes pour que ce champ d'activités soit déjà en plein essor.

C'est ainsi que l'application la plus élémentaire des machines à la linguistique est celle qui consiste à construire les *concordances* d'un texte ou d'un auteur donné, c'est-à-dire d'établir une liste des mots utilisés dans ce texte ou par cet auteur avec les références des endroits où ils sont utilisés. Citons à ce propos les travaux de l'équipe dirigée par J. Quemada à Besançon.

Dans le même esprit, l'on voit se multiplier les aides de la machine au traducteur humain, systèmes de compromis qui permettent d'attendre le développement sur une échelle plus grande, de la traduction automatique.

Il s'agit ici de programmes qui établissent des nomenclatures multilingues, des statistiques à usage stylistique, etc. Nous y avons fait allusion au cours du chapitre V. Nous nous contenterons de citer le système Dicautom, mis au point par L. Hirschberg à l'Université de Bruxelles.

La *documentation automatique.* - Dans les systèmes qui sont actuellement en fonctionnement, les documentalistes doivent encore caractériser les documents par un certain nombre de *mots clés,* de *relations* entre ces mots clés, etc. Il en va de même pour les questions qui leur sont posées. Mais, une fois ce travail accompli, le reste, c'est-à-dire le stockage de l'information, puis la recherche des documents qui répondent à une question, est entièrement automatisé. Il s'agit donc d'une aide de la machine au documentaliste. Mais déjà, on met au point des programmes d' « analyse automatique » pour la recherche des mots clés et même des relations. Ici encore, des langages spécialisés apparaissent, pour la formalisation de l'analyse comme pour la programmation. Nous citerons le « Syntol » de J.-C. Gardin et le « Comit » de Yngve.

Il faut également citer ici l'important domaine de l'enseignement automatique. Il s'agit essentiellement de la manipulation de données en langage naturel, avec correction automatique, etc. Bien entendu, la machine n'a pas à rechercher la solution correcte, les différents types de fautes ont été prévus. Il n'en reste pas moins qu'on est encore dans une situation de manipulation de données complexes.

Le dernier domaine que nous citerons ici comprend deux rubriques qu'il est assez naturel de rapprocher : le diagnostic automatique et la jurisprudence automatique. Dans les deux cas, il s'agit d'une aide de la machine à une profession libérale (médecin ou avocat). Dans les deux cas, il s'agit de manipuler des masses considérables d'informations rédigées principalement en langage naturel, avec peut-être l'aide de classifications documentaires du type mots clés, relations, et ainsi de suite. Dans les deux cas, enfin, il n'y a pas - du moins à notre connaissance - de réalisation complète au stade du fonctionnement quotidien.

Le problème du diagnostic automatique est d'établir, grâce à la possibilité de manipuler un stock d'informations énorme, des corrélations bien fondées entre système de symptômes et maladies (et d'en

déduire éventuellement un système de prescriptions). Au fond, c'est une variante du problème de la documentation automatique.

Dans le cas de la jurisprudence, il s'ajoute souvent au problème documentaire un problème de résolution d'équations logiques, puisque la législation se présente souvent sous la forme de décisions conditionnelles.

Des publications spécialisées et des congrès ont déjà rassemblé des chercheurs du domaine que nous venons d'évoquer. Il est clair que l'évolution de ces professions vers une structure en « cabinets » comprenant de nombreux collaborateurs permettra les investissements nécessaires au progrès de l'automatisation.

LA LITTÉRATURE ARTIFICIELLE

Nous aborderons maintenant un quatrième et dernier domaine d'application des machines, domaine qui a souvent excité l'intérêt (mais aussi l'imagination) des journalistes : c'est celui de la création.

Il y a déjà fort longtemps que J. Swift, dans les célèbres *Voyages de Gulliver,* décrivait une machine à composer de la littérature. Plus récemment, on a présenté des poèmes composés par des calculatrices. Que faut-il en penser ?

Il nous semble que la conception d'une littérature automatique (au sens d'une création littéraire réalisée par des automates et non dictée par l'inconscient comme l'essayèrent les surréalistes) est un peu prématurée.

Ce qui est abordable immédiatement, c'est l'aide de la machine au littérateur. En effet, de même qu'il existe des dictionnaires de rimes, des dictionnaires analogiques, des dictionnaires de citations, etc., il est concevable de développer des programmes qui se substituent à l'écrivain pour imposer aux textes qu'il manipule des contraintes plus complexes que les contraintes phonétiques de la prosodie, telles que le sont les contraintes de style, de situation, etc. Ce genre de considérations est d'ailleurs au centre des préoccupations de groupements comme l'Oulipo (OUvroir de LIttérature POtentielle) qui rassemblent écrivains et mathématiciens, groupements qui se situent par conséquent, eux aussi, aux frontières de l'intelligence artificielle.

En fait l'utilisation des machines en littérature peut se faire dans deux directions différentes.

D'une part on peut envisager une utilisation *analytique.* Cette utilisation est d'ailleurs amorcée dans les travaux de linguistique appliquée auxquels nous avons fait allusion au début du paragraphe précédent. On peut en effet étudier, non seulement la distribution des mots, le choix du vocabulaire, mais l'agencement des phrases, la fréquence des doubles, triples de mots revenant régulièrement, etc. Mieux, les programmes d'analyse syntaxique automatique peuvent mettre en évidence la préférence marquée d'un auteur pour tel type de construction, faire apparaître chez l'un des stemmas « en profondeur » et chez l'autre des stemmas «en largeur», bref on peut développer dans une certaine mesure une analyse stylistique automatique ou tout au moins semi-automatique.

L'utilisation des machines dans le domaine de la *synthèse,* c'est-à-dire de la création littéraire est évidemment plus délicate et peut donner lieu à de plaisantes anticipations (1). On peut par exemple utiliser des textes déjà existants et en créer de nouveaux à l'aide de substitutions systématiques (compte tenu de contraintes grammaticales et autres). Nous n'entrerons pas ici dans le détail de telles perspectives pour lesquelles on pourra se reporter avec fruit à ceux des travaux de l'Oulipo qui ont été déjà publiés (2).

LA MUSIQUE ARTIFICIELLE

Le domaine du langage est celui que nous avons abordé en dernier au cours de notre enquête sur les différents domaines de l'intelligence artificielle. Nous avons en effet constaté qu'il s'y combine la plupart des difficultés que l'on rencontre dans les autres domaines et ce n'est pas étonnant puisque c'est bien le langage qui sous-tend l'ensemble de nos activités et, finalement, de nos civilisations.

Ceci explique que, dans le domaine de la création et de sa simulation par les machines, l'aspect linguistique n'ait pas encore donné lieu à des développements bien importants.

(1) Comme dans l'excellent roman de R. ESCARPIT *Le littératron.*

(2) Cahiers du Collège de Pataphysique, dossier n° 17, 1961 ; voir aussi R. QUENEAU, *Bâtons, chiffres et lettres,* Gallimard, 1965, p. 317.

Par contre, la musique offre des possibilités beaucoup plus grandes pour l'utilisation des automates. On est en effet, en présence, ici, d'un système complètement formalisé et dont les aspects algébriques ont été mis en évidence depuis longtemps (par exemple dans le *Traité d'harmonie* de Paul Hindemith).

Dans le domaine de la musique, des efforts sérieux ont été accomplis dans l'esprit « automate » : la composition musicale peut en effet être considérée comme l'écriture de séquences de symboles compte tenu d'un certain nombre de contraintes (ton, mode, rythme, lois de l'harmonie, etc.).

Là, le nombre relativement restreint de conceptions à mettre en oeuvre permet de construire assez aisément des programmes pour automate. Par contre, les contraintes que l'on introduit alors semblent assez différentes de celles que l'on a vu apparaître au cours de l'évolution de la musique.

Après les travaux de Hiller et Isaacsan (1956) de nombreuses études ont été publiées (notamment en France par A. Moles, P. Barbaud et I. Xenakis). Il faut citer un travail remarquable du Soviétique R. Zaripov (1960). Il nous semble intéressant, sans reproduire l'organigramme d'ensemble, de donner la liste des sous-programmes qui le composent, avec une description grossière de leur contenu

A_o Formulation du programme en tenant compte de la mesure 3/4 ou 4/4.

A_1 Choix au hasard (ou commandé du pupitre) du caractère a du dessin rythmique qui dépend du nombre i (dans notre cas $i = 0, 1, 2, 3$).

A_2 Choix d'une combinaison rythmique j où j = o, 1, 2, 3 correspond aux combinaisons de valeur  admises dans le programme.

A_{31} Élimination des syncopes lorsque $j = 1$ et 3. Lorsqu'il y a une syncope, la transmission de l'ordre s'effectue en A_{33}. A la fin de la proposition le passage se fait sans condition en q_{46}.

A_{44} Inscription d'une combinaison rythmique.

A_{45} Formation des chiffres pris au hasard.

q_{45} Opérateur de transmission sans condition bII.
A la fin d'une proposition rythmique bII sur A_{5k} à la fin d'une proposition mélodique bII sur A . A l'intérieur d'une proposition bII sur A_{2ik} A_{6im}. Ici i_k et i_m prennent l'une des valeurs $i = 0$, 1, 3, 3 selon que $k = 0, 1, 2, 3$ et $m = o, 1, 2, ..., 7$.

$A_{50, 51, 52}$ Préparation du programme pour l'élaboration rythmique des propositions.

A_{53} Préparation pour appliquer la mélodie α 1 sur le rythme α.

A_{54} Choix au hasard (ou commandé du pupitre) du caractère α d'un dessin mélodique selon que $i = 0, 1, 2, 3$.

A_{61} Choix de l'intervalle pour la note suivante.

A_{70} A_{71} Détermination du signe de l'intervalle. Obtention de la hauteur de la note suivante. Choix du nombre de notes consécutives allant dans la même direction (ascendante ou descendante).

p_{72} Limitation de la hauteur dans l'ambitus entre *fa* de l'octave basse et *do* de la 3e octave. Lorsque les conditions ne sont pas réalisées l'ordre est transmis pour obtenir un autre intervalle.

p_{73} Exclusion des grands intervalles consécutifs dans la même direction au moyen de la transmission d'un ordre pour obtenir des intervalles plus réduits.

p_{74} Vérification de la fin de la proposition. A l'intérieur de la proposition impression de la note (A_{78}).

A_{75} A la fin de la proposition envoi de bII en q_{46} auprès de A_{8m}.

$p_{76}A_{77}$ Essai de fin de période et attribution à la dernière note de la période du degré I, III, V. S'il ne s'agit pas de la fin de la période mais de la fin de la proposition on passe en A_{78}.

A_{78} Impression de la note (codifiée).

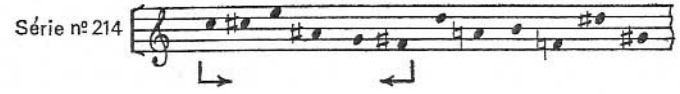
A_{8m} (m = o, 1, ..., 7) La préparation du programme pour l'élaboration mélodique de la proposition qui suit α.+).

q_{88} Arrêt (par un ordre donné du pupitre) ou retour automatique au début du programme (A_0) sans arrêt de la machine.

Ce qui nous semble très significatif, dans le domaine de la musique automatique, c'est l'effort entrepris par un compositeur comme A. Riotte, qui est aussi un spécialiste du calcul automatique, pour obtenir une aide de la machine au compositeur humain. Cette attitude nous semble beaucoup plus réaliste, tout au moins au stade actuel.

Riotte a utilisé la calculatrice pour engendrer des lignes mélodiques particulières qu'il a appelé « séries équilibrées ». Il s'agit de séries de douze sons dont les permutations possibles sont au nombre de l'ordre de 5.10^8, parmi lesquelles il y a seulement 519 séries équilibrées fermées.

A titre d'exemple, nous reproduisons l'une d'entre elles obtenue sur calculatrice I.B.M. 7090.



Nous ne pousserons pas plus avant dans le domaine des arts, afin de ne pas alourdir l'exposé. Le domaine des arts plastiques n'a pas encore été vraiment attaqué par les spécialistes des machines à calculer. Nous signalons seulement la notion d' « expert automatique » développée par J.-C. Gardin et nous-même en 1959 à l'occasion d'un problème de documentation archéologique.

* * *

Nous venons, dans ce chapitre, d'évoquer un nombre considérable d'activités qui se développent dans le cadre général de l'utilisation des machines à calculer. On se rend compte que ce sont là des activités d'intérêt souvent très immédiat et pour le progrès desquelles des sommes importantes sont investies. Or il est clair que ces activités sont étroitement liées à celles que nous avons rangées parmi les rubriques de l'intelligence artificielle. C'est dire qu'il existe là un ressort puissant pour la continuité d'un effort dans le domaine qui nous intéresse.

Certes des recherches intéressantes doivent pouvoir se poursuivre en l'absence de toute motivation pratique immédiate, mais lorsque de telles motivations existent, le chercheur peut envisager l'avenir avec plus d'optimisme notamment quand, comme c'est le cas ici, il a besoin d'utiliser des instruments qui coûtent très cher.

CHAPITRE VIII

Les idées et les hommes

Nous avons décrit l'extraordinaire prolifération des machines à calculer, et nous y avons trouvé la cause principale de cette pression nouvelle exercée dans les domaines les plus divers de l'activité intellectuelle.

Ce développement est, on s'en doute, le fait d'un nombre considérable de chercheurs. Même dans le cadre plus limité qui est celui de l'intelligence artificielle, les spécialistes se comptent aujourd'hui par centaines (dont la plus grande partie réside aux États-Unis).

Cependant il est clair que les germes d'un tel développement existent depuis longtemps. Comme c'est le cas pour toute grande idée, il est même possible de trouver des précurseurs dans le passé le plus lointain. Et c'est pourquoi, avant de présenter une sorte de bilan des oeuvres significatives et des hommes qui en sont les auteurs (et que nous avons, assez arbitrairement, répartis en trois classes : les précurseurs, les pionniers, les chercheurs), nous avons estimé nécessaire de rappeler la contribution de quelques grands ancêtres qui, bien avant que la première machine à calculer électronique soit mise en service, en avaient clairement conçu les possibilités et même, dans certains cas, prévu les applications.

DE LULLE A LEIBNIZ

C'est au Moyen Age que l'on voit se manifester pour la première fois l'une des tendances qui deviendra, beaucoup plus tard, caractéristique de l'âge des calculatrices : l'esprit combinatoire.

Cet esprit est en puissance dans la logique d'Aristote, mais c'est l'aspect particulier que prend l'enseignement péripatéticien dans la scolastique qui réalise vraiment cette potentialité.

Raymond Lulle (1232-1315) a illustré cette émergence de l'esprit combinatoire avec une netteté toute particulière. Au fond ce que propose Lulle dans son *Ars Magna* est un ensemble d'algorithmes (au sens moderne) permettant de résoudre une classe très vaste de problèmes. Il s'agissait de construire des assemblages de «catégories », propriétés », « vertus », etc. ; surtout il s'agissait de fournir une liste *exhaustive* des combinaisons possibles. Pour cela Lulle construisit des algorithmes de deux sortes, les uns purement graphiques, les autres contenant des éléments mécaniques (cercles concentriques pouvant effectuer des rotations l'un par rapport à l'autre).

Évidemment cette « mécanique » ne comporte aucune « intelligence» réelle. Bien plus, appliquée à des concepts mal définis, elle était utilisée à des démonstrations tantôt triviales, tantôt complètement illusoires.

Cependant, elle fut à l'origine du premier travail original de Leibniz et orienta profondément sa pensée.

Leibniz (1646-1716) est, par excellence, le grand ancêtre de l'intelligence artificielle. Tous les problèmes que nous avons évoqués au cours des chapitres précédents ont été abordés par lui et souvent avec une pénétration qui semble presque être prémonition !

Le cheminement même de sa pensée est instructif : elle commence par une réflexion sur le calcul des syllogismes avec un effort de formalisation des figures, modes, etc. Puis, l'étude de Lulle et de ses successeurs Kircher (1669), Dalgarno (1661), le conduit à l'art combinatoire, puis à la considération d'une langue universelle, enfin à la caractéristique universelle. Ce cheminement est celui d'une recherche de *symboles* et de *règles pour la manipulation* des *symboles* qui permettent d'exprimer les choses et les concepts de façon adéquate et de réduire les raisonnements auxquels ils donnent lieu à une manipulation purement mécanique des symboles.

Laissons la parole à Leibniz lui-même :

« C'est le but principal de cette grande science que j'ay accoustumé d'appeler *Caractéristique,* dont ce que nous appelons l'Algèbre n'est qu'une branche fort petite. Car c'est la Caractéristique qui donne les

paroles aux langues, les lettres aux paroles, les chiffres à l'Arithmétique, les notes à la Musique ; c'est elle qui nous apprend le secret de fixer le raisonnement, et de l'obliger à laisser comme des traces visibles sur le papier en petit volume, pour estre examiné à loisir c'est enfin elle qui nous fait raisonner à peu de frais, en mettant des caractères à la place des choses pour des-embarrasser l'imagination » (1).

Toute l'activité créatrice de Leibniz est contenue dans ce programme gigantesque, la *Monadologie* comme le calcul infinitésimal, mais aussi des projets plus spécifiques comme celui de machines à calculer. Après les machines arithmétiques et algébriques inventées dans sa jeunesse, il imagine une « grammaire cylindrique » qui fournirait tous les théorèmes appartenant à un système formel donné et envisage même l'application de ce mécanisme à la résolution de problèmes de droit !

Enfin Leibniz distingue clairement l'intérêt du problème des jeux comme exemple de problème combinatoire. Il est le premier à faire la distinction des jeux où le hasard intervient par opposition à ce que nous avons appelé les jeux « à information complète » (2).

On voit donc que l'ensemble de notre domaine est couvert, bien entendu de façon parfois superficielle. La combinatoire est commencée (notamment sous son aspect numérique : l'analyse combinatoire), mais la caractéristique demeure un projet. C'est un projet que notre siècle aborde seulement et que le suivant réalisera peut-être.

DE BABBAGE A JEVONS

Ch. Babbage (1792-1871) est bien connu comme l'inventeur de la première véritable machine à calculer complexe. Les machines inventées par Pascal, Leibniz et d'autres étaient les ancêtres de nos machines de bureaux, c'est-à-dire qu'elles étaient capables d'effectuer les opérations élémentaires de l'arithmétique.

Babbage au contraire mit au point (et réalisa en partie) une machine

(1) Cité par L. COUTURAT dans son ouvrage *La logique de Leibniz*, réédition G. OLMS, 1961.

(2) *Loc. cit.*, p.581

qui contenait les organes de mémoire et l'organisation logique des machines à calculer modernes. Mais, faute de disposer de techniques suffisamment souples (en fait seules les techniques électroniques devaient apporter plus tard cette souplesse), la réalisation pratique de la machine fut un échec.

Mais il est très intéressant de remarquer que les intérêts de Babbage le portaient bien au-delà du seul calcul automatique. Il avait en effet consacré beaucoup de temps et d'efforts à la mise au point de concepts qui deviendront ceux de la recherche opérationnelle. Bien plus, il avait étudié soigneusement la formalisation et la simulation par un automate de certains jeux, notamment du Tic-Tac-Dou auquel nous avons fait allusion dans le chapitre III. Pour toutes ces raisons, Babbage est incontestablement un ancêtre de l'intelligence artificielle.

S. Jevons (1835-1882) est encore un personnage captivant, aux intérêts variés et aux activités multiformes. Chimiste, économiste et logicien, il voit clairement, lui aussi, l'importance du choix d'une symbolique adéquate et la possibilité d'utiliser un mécanisme pour la manipulation automatique des symboles. Jevons, dans son article intitulé « Réalisation mécanique de l'inférence logique » (1), se réfère explicitement à Lulle et à Babbage. Les dispositifs mécaniques qu'il imagine sont directement liés à la nature particulière des problèmes qu'il veut traiter (calcul des propositions et calcul des prédicats du 1er ordre). Il se propose d'ailleurs essentiellement d'illustrer, grâce à cette machine, les possibilités du calcul inférentiel développé quelques années auparavant par G. Boole. Mais il est bien conscient de la généralité des méthodes utilisées et de l'étendue de leurs applications.

LES PRÉCURSEURS

Nous citerons ici les auteurs contemporains qui, à un titre ou à un autre, ont contribué au développement de l'intelligence artificielle avant même que les premières expériences sur calculatrices électroniques aient eu lieu.

(1) Publié en 1870 par les *Philosophical Transactions,* t. 16o, pp. 497-518.

Parmi ceux qui ont contribué au développement des notions mathématiques et logiques qui devaient être utilisées par la suite, nous citerons

G. Polya qui, par ses travaux d'analyse combinatoire (cités dans l'ouvrage de Beckenbach porté plus loin dans la « bibliothèque de base ») et ses recherches sur l'heuristique mathématique (1), apporte une double contribution à notre domaine.

C. Shannon a contribué de façon décisive au démarrage de la théorie de l'information (2). Il a joué aussi un rôle important dans la théorie des automates (3) et nous le retrouverons comme pionnier dans le domaine de la simulation des jeux.

A. Turing a introduit la notion de machine abstraite (4) puis a contribué notablement aux premières décisions sur l'intelligence artificielle (5). Il s'est aussi occupé de la simulation des jeux.

A la frontière de la logique et de la linguistique, citons les travaux de Y. Bar-Hilell et R. Carnap sur la notion d'information sémantique (6) ainsi que le rapprochement opéré entre syntaxe structurale et théorie des automates par N. Chomsky (7).

Enfin le problème général des automates a été initié par des recherches variées mais convergentes de R. Ashby (8), D. Mac Kay (9), W. McCulloch et W. Pitts (10), W. G. Walter (11) et, naturellement, N. Wiener (12).

A cela il convient encore d'ajouter les travaux multiples de J. von Neumann et notamment le livre cité dans la bibliographie finale.

(1) G. POLYA, *Induction and Analogy in Mathematic,* Princeton U. P., 1954.
(2) C. SHANNON-W. WEAVER, *The Mathematical Theory of Communications,* Un. of Ill. Press, 1949.
(3) C. SHANNON et J. MCCARTHY, *Automata Studies,* Princeton U. P., 1956.
(4) *Proc. of the London Math. Soc., 42,* 230, 1937.
(5) Cf. dans le livre de FEIGENBAUM et FELDMAN Cité plus loin, p. 11.
(6) Cf. *British ,1. of Philosophy of Science, 4,* 127, 1953-
(7) Cf. *T. of Symbolic Logic, 18, 247,* 1953-
(8) *Cf. Design for a Brain,* J. Wiley, 1952.
(9) *Cf. British T. of Philosophy of Science, 2,* 105, 1951.
(10) *Cf. Bull. of Mathematical Biophysics, 7,* 89, 1943.
(ri) The Living Brain, Norton, 1953.
(11) Cf. *Cybernetics,* Hermann, 1946.

On remarquera que les travaux que nous venons de citer sont tous antérieurs à 1955. Les travaux des «pionniers » ne commencent guère avant 1957 et sont publiés pour la plupart entre 1957 et 1959.

LES PIONNIERS

Les premières machines à calculer électroniques furent construites à la fin des années 4o dans des organismes publics aux U.S.A. et en Angleterre. Ainsi furent-elles utilisées essentiellement comme calculatrices, pour résoudre des problèmes précis et urgents. Ce n'est qu'avec la commercialisation des calculatrices que l'on voit apparaître, dans les universités et même chez les constructeurs eux-mêmes (notamment à I.B.M.), des applications de caractère plus aventureux comme le sont celles liées aux problèmes de l'intelligence artificielle.

Dans le domaine des jeux, il faut citer les recherches de A. Bernstein, A. Turing, C. Shannon, S. Ulam, puis le grand travail de A. Newell, J. C. Shaw et H. Simon (i), enfin les recherches de J. McCarthy et, en U.R.S.S., celles de Shura-Bura.

Tous ces travaux portaient sur le jeu des échecs. Le jeu de dames fut étudié avec succès par A. Samuel (2).

Dans le domaine de la démonstration automatique des théorèmes, il faut citer E. Beth, M. Davis, H. Gelernter, P. Gilmore, et de nouveau Newell, Shaw et Simon. L'auteur le plus important est H. Wang (3).

Dans le domaine linguistique, les pionniers se nomment L. Dostert, S. Ceccato, A. Oettinger, V. Yngve, O. Kulagina, M. Masterman, I. Melchuk, D. Hays, en ce qui concerne la traduction automatique, P. Luhn, C. Mooers, M. Maron, A. Parker-Rhodes, M. Taube et J. C. Gardin en ce qui concerne la documentation automatique.

Pour des applications variées il faudra citer les noms de A. Andrew, H. Borko, E. Feigenbaum, J. Feldman, R. Friedberg, B. Green, M. Kochen, G. Pask, N. Rochester, R. Solomonoff, F. Tonge, etc.

(1) Cf. I.B.M. *Journal of Research and Developrnent, 2,* 320, 1958.
(2) Cf. I.B.M. *Journal of Research and Developrnent, 3,* 210, 1959.
(3) Cf. I.B.M. *Journal of Research and Developrnent, 4,* 2, 1960.

Les hérauts de l'intelligence artificielle à cette époque déjà reculée (c'est-à-dire comprise entre 1955 et 1960 !) sont M. Minsky (1) et O. Selfridge aux États-Unis, D. Lyapunov en U.R.S.S. *(2)* et F. Le Lionnais en France (3).

LES CHERCHEURS

Ils sont légion aujourd'hui et il n'est pas question ici de les dénombrer. Il nous a paru cependant utile de donner une sélection de quelques noms à titre de guide pour le lecteur désireux de suivre de plus près des orientations particulières de la recherche.

Dans la période la plus récente, la simulation des jeux n'a pas été poussée très loin, en raison des grandes difficultés rencontrées pour les échecs. Seuls M. Euwe et ses collaborateurs ont déployé une activité qui, quoique fort intéressante, n'a pas pu être menée à son terme (4). Par contre, d'autres jeux ont été abordés : le Go par A. Remus, le Go-Bang par P. Braffort, A. Lussan et H. Van Hedel.

Le domaine le plus populaire a été celui de la linguistique avec P. Garvin, H. Hiz, D. Lehman, S. Pendergraft, Y. Lecerf, L. Hirscherg, S. Greibach, G. King, S. Kuno pour la traduction automatique; L. Doyle, A. Leroy, R. Simmons, V. Giuliano, R. Needham, P. Edmundson et G. Salton pour la documentation automatique.

C'est le domaine de la démonstration automatique des théorèmes qui a été le plus actif et dans lequel les progrès ont été les plus significatifs.

Outre les « pionniers » cités plus haut et qui, pour la plupart, ont continué à contribuer aux recherches, de nouveaux talents sont intervenus, parmi lesquels nous citerons D. Prawitz, J. Slagle, A. Robinson, J. Robinson, L. Wos, G. Collins, B. Dunham, J. Friedman, S. Kanger, D. Lehmer, S. van Westrhenen.

(1) Sa bibliographie sur l'intelligence artificielle reste un instrument de travail indispensable. Elle a été réimprimée dans le livre édité par FEIGENBAUM et FELDMAN, cité dans la « bibliothèque de base » en fin de chapitre.
(2) Grâce à la publication de la revue *Problèmi kybernetiki.*
(3) Voir le chapitre consacré à la cybernétique dans *l'Histoire générale des Sciences,* Presses Universitaires de France.
(4) Plus récemment, J. McCarthy et ses collaborateurs ont repris et amélioré leur programme et sont entrés en compétition avec leurs collègues soviétiques.

Pour des applications diverses, indiquons, les noms de S. Amarel, F. Black, D. Bobrow, R. Kirsch, P. Armer, R. Lindsay, etc.
Bien entendu ce petit *Who's who* de l'intelligence artificielle n'est ni complet ni définitif. Chaque année de nouveaux chercheurs entrent en lice tandis que certains de leurs aînés passent à d'autres domaines d'activité. On peut dire qu'après une phase de développement un peu chaotique à la fin des années 5o et au début des années 6o, développement lié à une publicité tapageuse et mal informée (demain on traduira gratis !), un certain reclassement s'est opéré : tassement du nombre des publications mais amélioration sérieuse de leur qualité. Après la ruée vers l'or et les déceptions qui suivirent, l'heure est maintenant au travail de laboratoire.

UNE BIBLIOTHÈQUE DE BASE

Le lecteur désireux d'approfondir les questions traitées dans cet ouvrage devrait posséder des connaissances, au moins élémentaires, dans le domaine des machines électroniques. Celui qui ne les possède pas pourrait utilement s'initier à l'aide des livres suivants :

P. DEMARNE et M. ROUQUEROL, *Les ordinateurs électroniques,* Presses Universitaires de France, coll. « Que sais-je ? », n° 832.

B. RENARD, *Le calcul électronique,* Presses Universitaires de France, coll. « Que sais-je ? », n° 882.

J. et J. PoYEN, *Le langage électronique,* Presses Universitaires de France, coll. « Que sais-je ? », n° 900.

On trouvera un exposé plus approfondi, plus audacieux aussi dans ses perspectives, dans :

F.-H. RAYMOND, *L'automatique des informations,* Masson, 1957.

L'aspect historique est plus particulièrement développé dans les ouvrages suivants :

M. GARDNER, *L'étonnante histoire des machines logiques,* Dunod, 1965.

E. C. BERKELEY, *Cerveaux géants, machines qui pensent,* Dunod, 1963.

F. GILLOT, *Algèbre et logique* d'après les textes originaux de G. BOOLE et W. S. JEVONS, Blanchard, 1962.

J. BERNSTEIN, *The Analytical Engine,* Sicker & Warburg, 1964.

Les problèmes mathématiques liés aux questions que nous avons soulevées sont traités dans des ouvrages assez spécialisés, cependant on en trouve l'essentiel sous une forme remarquablement accessible dans :

B. A. TRAHTENBROT, *Algorithmes et machines à. calculer,* Dunod, 1963.

Pour les lecteurs munis d'un bagage mathématique déjà important, on doit signaler l'ensemble des articles rassemblés dans :

D. BECKENBACH (ed.), *Combinatorial Mathematics,* J. Wiley, 1964.

Le premier livre directement lié au sujet principal qui nous préoccupe est :

P. de LATIL, *Introduction à la cybernétique : La pensée artificielle,* Albin Michel, 1953, mais l'accent reste mis sur le côté « cybernétique » des problèmes.

L'auteur analyse les différents aspects de la « rétroaction », etc.
C'est seulement après 1960 que, sous l'impact des publications de la presse spécialisée et notamment après le congrès de l'I.F.I.P. tenu à Paris en 1959, les ouvrages se multiplient : ouvrages originaux ou collection de réimpression des textes fondamentaux.
En voici la liste :

H. BoRKo (ed.), *Computer Applications in the Behavioral Science,* Prentice Hall, 1962.

M. GREENBERGER (ed.), *Management of the Computer of the Future,* N.I.T. Press, 1962.

M. TAUBE, *Computer and Common Sense,* McGraw-Hill, 1963.

P. BRAFFORT et D. HIRSCHBERG, *Computer Programming and Formal Systems,* North Holland, 1963.

S. GARVIN, *Natural Language and the Computer,* McGraw-Hill, 1963.

E. A. FEIGENBAUM et J. FELDMAN, *Computers and Thought,* McGraw-Hill, 1963. (Ce livre est probablement le plus important ; il contient des articles fondamentaux relatifs aux questions traitées dans les chapitres III, IV et VIII.)

M. S. PEDELTY, *An Approach to Machine Intelligence,* Spartan Books, 1963.

SAYNE et CROSSON, *The Modeling of Mind,* Notre-Dame Univ. Press, 1963.
Tou et WILCOX, *Computer and Information Sciences,* Spartan Books, 1964.
M. A. SASS et W. D. WILKINSON, *Computer Augmentation of Hu man Reasoning,* Spartan Books, 1965.
D. FINIC, *Computers and the Human Mind,* Doubleday, 1966.
Fox, *Advances in Programmning and Non-Numerical Computation,* Pergamon, 1966.
P. GARVIN, *Computation in linguistics,* Illinois U. P., 1966.
M. VALACH, *Cybernetic modelling,* Iliffe, 1967.
BooTH, *Machine translation,* North Holland, 1967.

Et, bien entendu, les numéros spéciaux consacrés aux calculatrices :
Le numéro de *Scientifc American* de novembre 1966, avec l'article de M. MINSKY sur l'intelligence artificielle, et le numéro de janvier 1967 des *Proceeding of the LE.E.E.,* avec l'article de SOLOMONOFF sur le même sujet.
Il faut encore ajouter à cette liste un ouvrage exceptionnel par sa largeur de vue et la profondeur de ses implications :

J. von NEUMANN, *The Computer and the Brain,* Yale Univ. Press, 1958;

ainsi qu'un recueil de discussions souvent philosophiques mais toujours fort intéressantes :

S. HOOK (ed.), *Dimensions of Mind, N. Y.* Univ. Press, 1960.

Épilogue

Il faut maintenant conclure. Nous avions, dans le chapitre II, introduit un langage et exprimé dans ce langage, au cours des chapitres suivants, le problème de l'intelligence artificielle. Puis, dans le chapitre VI, nous avions mis l'accent sur leur caractère commun : la recherche de procédures anticombinatoires pour réduire la complexité, à divers niveaux.
Enfin, dans les deux derniers chapitres, nous avons décrit l'environnement de l'intelligence artificielle. Nous croyons avoir montré ainsi que le développement des applications récentes des calculatrices amènerait au niveau de la nécessité la résolution de problèmes que les recherches du type « démonstration automatique des théorèmes » n'auraient pas pu faire sortir du niveau de la recherche gratuite. Nous avons indiqué aussi qu'à l'audacieuse brigade des pionniers un fort contingent de jeunes chercheurs s'était joint récemment et se mettait au travail avec une ardeur accrue.
Nous pouvons en conclure, tenant compte aussi des développements technologiques auxquels il faut s'attendre, que *l'ère de l'intelligence artificielle est commencée.*
Y a-t-il lieu d'en tirer des conséquences d'ordre moral ou métaphysique ? Nous ne le pensons pas.
Nous avons soigneusement évité, d'un bout à l'autre de cet ouvrage, d'établir un quelconque parallélisme entre le fonctionnement des automates et celui des cerveaux (humains ou non). Visiblement les techniques utilisées sont différentes et si les « programmes » - qui demeurent inconnus dans le cas du cerveau - devenaient identifiables, il n'y aurait rien de plus à en conclure que de la similitude de structure du bras humain avec un levier.

Nous avons d'ailleurs écarté volontairement de notre propos l'utilisation des machines pour les études de psychologie du comportement, bien que de telles études fournissent l'essentiel de la copie d'une revue comme *Behavioral Science.*

Au fond, l'attitude de tant de bons auteurs qui (en France, aux U.S.A., en U.R.S.S.) repoussent avec indignation l'éventualité d'une automatisation des activités intellectuelles supérieures, relève de ce que nous proposons d'appeler un « *complexe de Frankenstein* ». P. Armer et M. Minsky ont souvent cloué au pilori ceux de leurs compatriotes qui, comme M. Taube, s'efforcent d'arrêter les recherches ou de les déconsidérer.

On trouve l'attitude inverse chez certains enthousiastes comme I. Good qui, dans un article récent intitulé « Speculations Concerning the First Ultraintelligent Machine », n'hésite pas à déclarer : « La survie de l'espèce humaine dépend de la mise en construction rapide d'une machine ultraintelligente. »

Notre position dans ce débat sera, nous le craignons, très banale pour nous l'intelligence artificielle est un champ de recherches qui s'est constitué en prolongement naturel à d'autres recherches et continuera de se développer d'autant plus vite que la pression de la demande se fera davantage sentir.

Les cris et les anathèmes n'y feront rien, pas plus que l'obstination des énergétistes n'a empêché le développement et les succès de la physique des atomes et des particules.

Ce qui arrive parfois, c'est que, sous l'influence d'une idéologie désuète ou mal assimilée, certains savants de valeur se privent délibérément de la possibilité de contribuer au progrès de la science.

Que, dans le cas des automates, les progrès de la science soient ou non bénéfiques à l'humanité, c'est une question à laquelle nous n'avons pas les moyens de répondre et que, peut-être, il n'y a pas de sens à poser.

Table des matières

1968-2 - Imprimerie (les Presses Universitaires de France. - Vendôme (France)
ÉDIT. N° 29 878 IMPRIMÉ EN FRANCE IMP. N° 20 754