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.

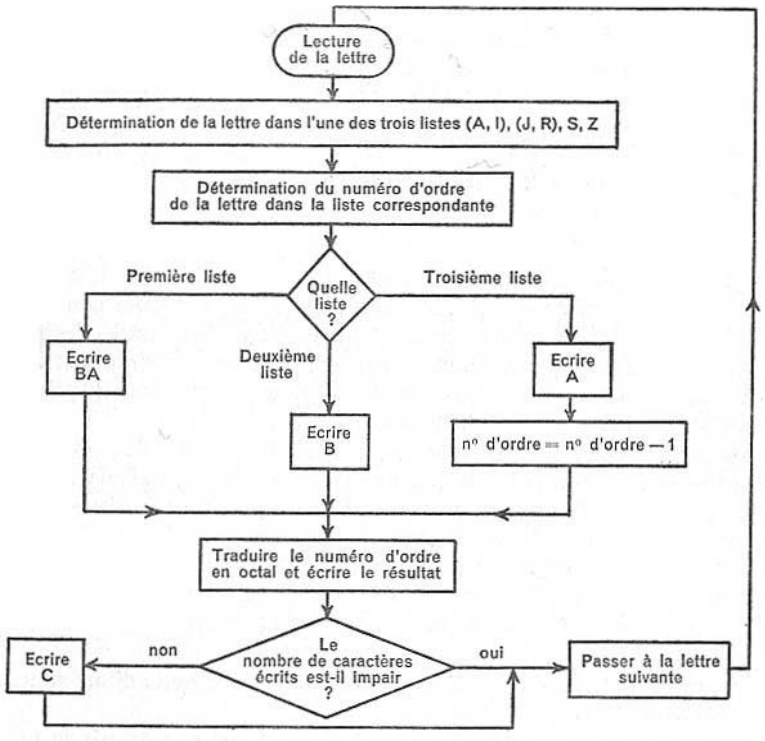
Au cours du chapitre III, nous avons illustré cette thèse en analysant quelques jeux, dont la simulation a été tentée sur calculatrice (et en particulier l'un d'entre eux, le jeu de Go-Bang). Nous avons observé alors que la simulation des jeux nécessitait tout d'abord une double codification : celle 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

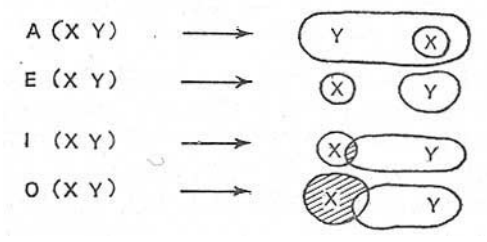


FIG. 27. - Mise en correspondance des prédicats traditionnels du syllogisme avec une représentation ensembliste
On remarquera que les parties hachurées sont nécessairement non vides

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_0, (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 $H_0 \cup H_1$, 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.

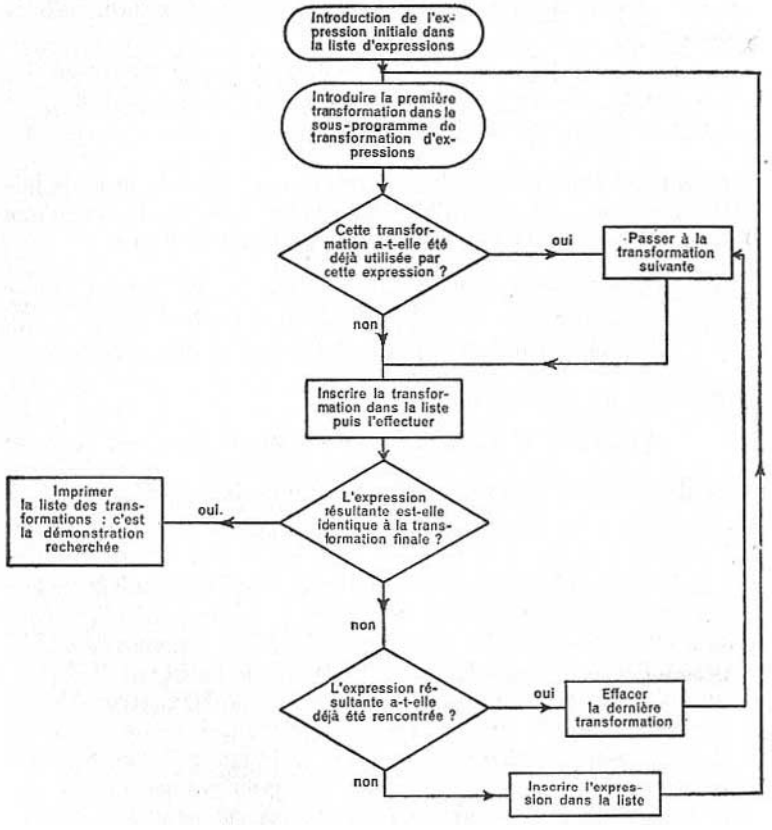


FIG. 28. - Organigramme correspondant à la recherche d'une démonstration automatique de la formule $2^m + 2^n = 2^{(m+n)}$

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

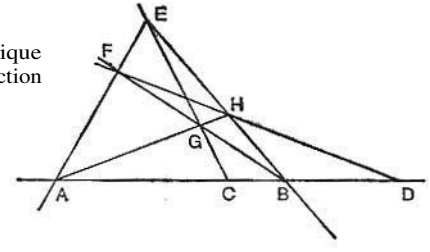


FIG. 29. - Figure géométrique correspondant à la construction

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

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

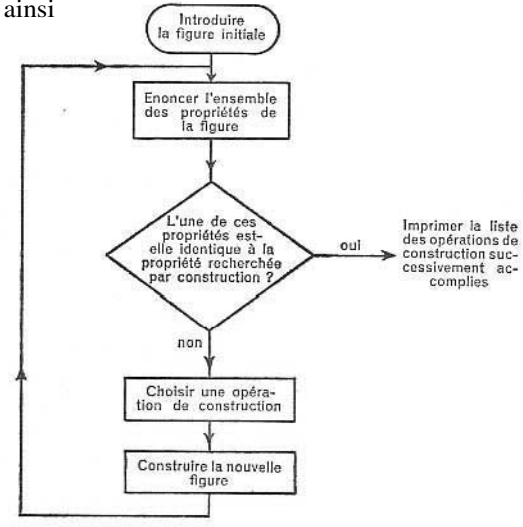
La conjugaison harmonique de deux couples de points (P_1, P_2), (P_3, P_4) s'écrit .

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

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