Parler avec les machines

André Lentin et Louis Nolin

Avant-Propos. – Cet article est paru initialement dans la revue *ATOMES* en avril 1968 au numéro 253. La revue *ATOMES* qui parut pour la première fois en mars 1946 et continua sa route jusqu'en avril 1970, puis prit le nom de *La Recherche*.
Tous les articles de *ATOMES* ne concernent pas que la Physique, loin de là, mais aussi la médecine, la chimie, l'électronique, la mécanique, etc. Prenons exemple du numéro 1 de *ATOMES,* où un article de Sir Howard Walter Florey traite de la Pénéciline.
Revenons au numéro 253 de la revue, André Lentin et Louis Nolin sont en compagnie, entre autres, d'André Warusfel, grand vulgarisateur et pédagogue des mathématiques.
Son article aborde « Scientifiques » et « littéraires » sur les ondes.
Denis Glaizot, passionné de sciences et techniques, administrateur et rédacteur en chef de Gloubik Sciences, nous a retrouvé cet article historique et autorisé avec enthousiasme à le diffuser au format électronique. Nous avons remis en forme l'exemplaire original. Denis Glaizot est l'une des « plumes » du site (https://sciences.gloubik.info/) avec Lydie Blaizot. Nous le remercions vivement.
Nous rappelons qu'André Lentin et Louis Nolin sont aussi des « plumes » historiques de notre périodique. (N.D.L.R.)

---0---

■ *Le développement actuel du calcul automatique dans différents secteurs de l'économie et de la recherche pose de délicats problèmes. Ceux-ci ne sont pas toujours évidents à priori : c'est l'expérience qui les a généralement révélés et c'est elle aussi qui permet souvent de leur apporter une solution empirique au moins partielle. Il est très difficile, en effet, d'essayer de les résoudre par une démarche théorique qui, pourtant, serait évidemment souhaitable, ne serait-ce que pour permettre une meilleure utilisation du matériel.*

■ *Certaines des difficultés rencontrées proviennent de la distance très grande séparant l'homme de la machine qui effectue pour lui les calculs auxquels elle est dressée. Si une communication homme-machine est possible (bien que cette expression soit presque aussi anthropomorphique, et donc contestable, que celle de « cerveau » électronique), elle ne peut s'instaurer qu'à travers des codes très particuliers. Les langages, sans cesse évolutifs, tendent à devenir de plus en plus des objets d'étude ayant leur intérêt propre en dehors de leur utilisation pratique immédiate. La structure algébrique qui les sous-tend est d'une très grande simplicité, que ne laisse guère entrevoir le formalisme rigide auquel doivent se plier programmateurs et analystes.*

Qu'est-ce qu'une machine ? Cette question peut et même doit être posée de plusieurs façons distinctes.

Bien entendu, une machine c'est d'abord, ici, un assemblage électronique nanti d'un certain nombre de connexions et qui fonctionne d'une certaine façon. Tout ce qu'on peut lui demander, c'est d'être capable d'effectuer de façon satisfaisante un petit nombre d'opérations élémentaires. On distingue les machines analogiques et les machines digitales. Sur le plan de la logique de fonctionnement aussi bien que du point de vue de l'électronicien, cette distinction est utile : cependant, ce sont toujours des signaux qui sont traités dans les deux cas, non des nombres.

Une machine digitale, électroniquement parlant, comporte essentiellement, outre les entrées et les sorties qui sont les organes de communication, des unités de mémoire et une unité de calcul, centre nerveux de la machine. Les nombres sont mis en mémoire sous forme d'états internes, qui sont les polarisations de tores de ferrite. On symbolise par 0 ou 1 les deux états possibles.

Inutilisable toute nue.

Sans nous appesantir sur la technique de fonctionnement de la machine, signalons cependant les éléments dont elle a besoin pour fonctionner :

- *Les opérations*. -- Elles sont en petit nombre, identifiées par un signal qui correspond à un numéro. Cette propriété permet de mettre en mémoire les instructions. Toutes les machines comportent l'addition, la soustraction, la comparaison, les échanges avec l'extérieur, l'arrêt. D'autres opérations (négation, produit booléen, etc.) existent sur la plupart des machines. Depuis la calculatrice de bureau jusqu'au véritable ordinateur, toutes les gammes de complexité existent.
- *L'adressage*. -- Il ne suffit pas d'avoir en mémoire une série de nombres, il faut savoir où trouver quoi. Pour cela il est nécessaire de repérer par un système de coordonnées les différentes parties de la mémoire et de disposer d'accès spécifiques à celles-ci.

Nous voyons donc que, pour effectuer une séquence de calcul telle que 2 + 3, nous devrons lire 2, lire 3, effectuer sur ces deux nombres l'opération n°5 (par exemple) qui est le code de l'addition, et écrire le résultat. Si cela se présente dans une séquence de calcul, il nous faudra 1° le code d'opération, 2° les adresses respectives où l'on trouve les nombres sur lesquels l'opération sera effectuée, 3° la destination (adresse) où ranger le résultat.

Respectivement spécialistes de logique mathématique et d'algèbre, les mathématiciens André LENTIN et Louis NOLIN sont associés, depuis une dizaine d'années, au professeur DE POSSEL à la direction de l'institut Blaise-Pascal (CNRS). Ils ont été parmi les premiers à développer les aspects théoriques de l'informatique tout en liant cet effort aux multiples applications rencontrées dans la pratique. Ils sont coauteurs, avec MM. ARSAC et NIVAT, d'un manuel d'Algol.

Un certain nombre de conventions permet de simplifier la procédure pratique, mais la base du fonctionnement reste celle-là.

Il faut insister sur le fait qu'une machine « nue », c'est-à-dire non assortie d'un ensemble de programmes de base, est à peu près inutilisable. Pour communiquer avec elle, on dispose d'une centaine d'instructions élémentaires (codées en binaire) dont chacune correspond à une opération simple (par exemple : transférer le contenu d'une cellule de mémoire à un certain registre). Pour réaliser une opération plus complexe, on peut utiliser un agrégat d'instructions élémentaires, un microprogramme, qu'on laissera à demeure dans la mémoire centrale. Pour faire fonctionner la machine, en particulier pour la mettre en relation avec ses périphériques d'entrée et de sortie, il faut utiliser de nouveaux programmes, plus longs, qu'on pourra laisser dans les mémoires auxiliaires, etc. En d'autres termes, et dans l'état actuel de la technique, il s'avère indispensable d'ajouter, aux instructions et opérations câblées du « hardware », tout un «software » d'instructions et opérations *programmées*.

Le grand ancêtre : la machine de Turing

C'est uniquement par abus de langage que l'on continue à appeler *machine informatique* l'objet complexe formé par une machine et un assortiment de programmes de base logés dans la mémoire. Que cet abus de langage puisse ou non être défendu sur des bases théoriques, il comporte un risque d'erreur pour le non-initié.

Pour le mathématicien, ces considérations n'ont guère d'intérêt. Peu lui importe au fond que, dans une machine, il n'y ait pas de nombres mais des aimants et des signaux carrés : ce qui compte pour lui, c'est que le fonctionnement de la machine ait des propriétés stables et cohérentes qu'il posera suivant un modèle abstrait et axiomatisé. Dans ce domaine, il faut même remarquer que la théorie a devancé la pratique : dès 1936, le logicien Turing avait envisagé une machine abstraite où l'on peut voir la préfiguration, très stylisée, des machines réelles.

Il s'agissait, à l'époque, de trouver un concept mathématique propre à formaliser de façon adéquate la notion intuitive de *calculabilité* (indépendamment de toute contrainte pratique d'espace et de temps). Turing dégagea un tel concept et le présenta sous la forme d'une « machine ». Cet automate dispose d'une bande, potentiellement infinie, divisée en cases, sur laquelle il peut écrire et (effacer) en usant d'un alphabet fini à raison d'un symbole par case. L'organe de calcul est capable de prendre un nombre fini d'états internes. Une situation de la machine est le couple formé par le symbole lu a_i et l'état S_j au cours duquel se fait la lecture. A partir d'une telle situation, la machine évolue en passant à un état S_l et en faisant un acte qui peut être :

- écrire un *certain* a_k, à la place de a_i,
- avancer d'une case (ce qu'on symbolise par $\rightarrow$)
- reculer d'une case (ce qu'on symbolise par $\leftarrow$)

La liste des quadruplets du type

$$a_i \quad S_j \quad a_k \quad S_l$$
$$a_i \quad S_j \quad [\rightarrow] \quad S_l$$
$$a_i \quad S_j \quad [\rightarrow] \quad S_l$$

caractérise la machine. Le fonctionnement en est déterministe, car un seul quadruplet (au plus) répond à une situation donnée. Le calcul d'une machine de Turing commence par l'examen dans un état initial du premier symbole de la donnée. Il peut durer indéfiniment. Si la machine s'arrête, le contenu de la bande donne le résultat correspondant à la donnée.

Toutes les définitions de la calculabilité que l'on a proposées se sont révélées équivalentes à la Turing-calculabilité : est calculable que ce que peut calculer une machine Turing.

En particulier les calculatrices réelles correspondent aux machines de Turing dites universelles ; munies d'un programme, elles calculent la semi-fonction Turing calculable que caractérise le programme.

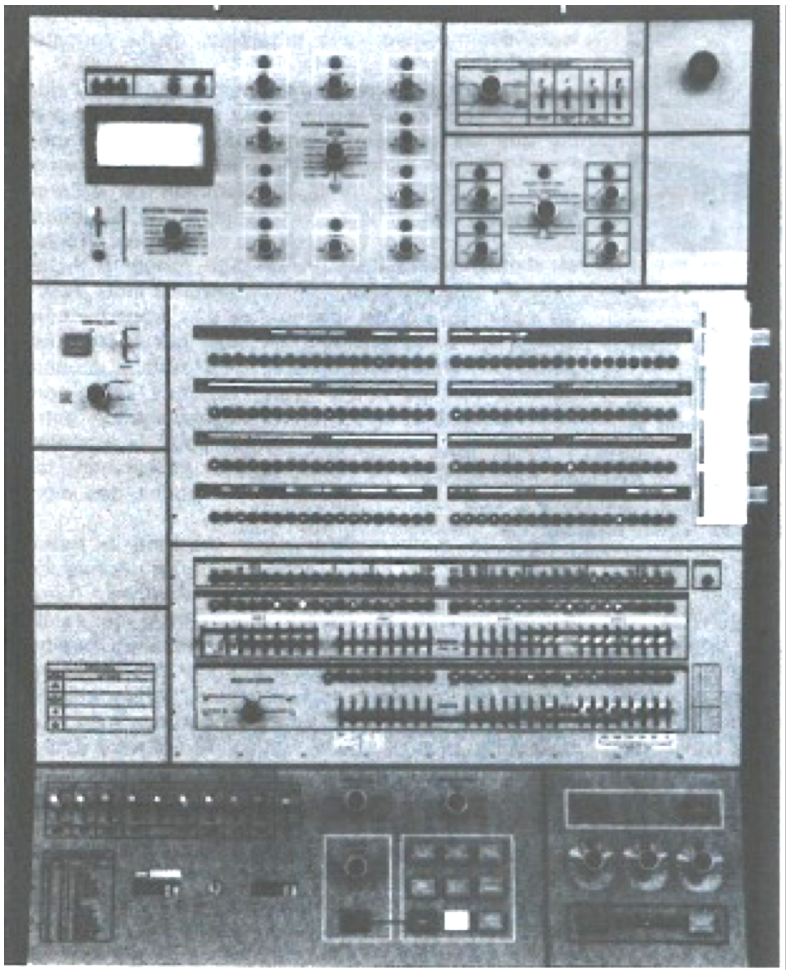
Par un juste retour des choses, la pratique a, par la suite, alimenté la théorie en problèmes et a conduit à de nouveaux concepts.

Certains organes de machines correspondent à la notion d'automates *finis*. L'une des réalisations possibles d'un automate abstrait fini conduirait à prendre une machine de Turing qui ne peut que lire son ruban et avancer. Dans la hiérarchie des automates, il occupe la place la plus basse, la plus haute étant réservée aux machines de Turing.

Entre les deux, il y a toutes les variétés d'automates à mémoire linéairement bornée. Celui qui a suscité le plus d'études est l'automate « push-down ». Il possède deux bandes, l'une de donnée, l'autre de travail, avec ces restrictions qu'il ne peut que progresser dans sa donnée et que, s'il recule sur la bande de travail, il efface la partie correspondante et perd l'information qu'elle contient. L'automate push down formalise la notion de *pile*, bien connue des programmeurs.

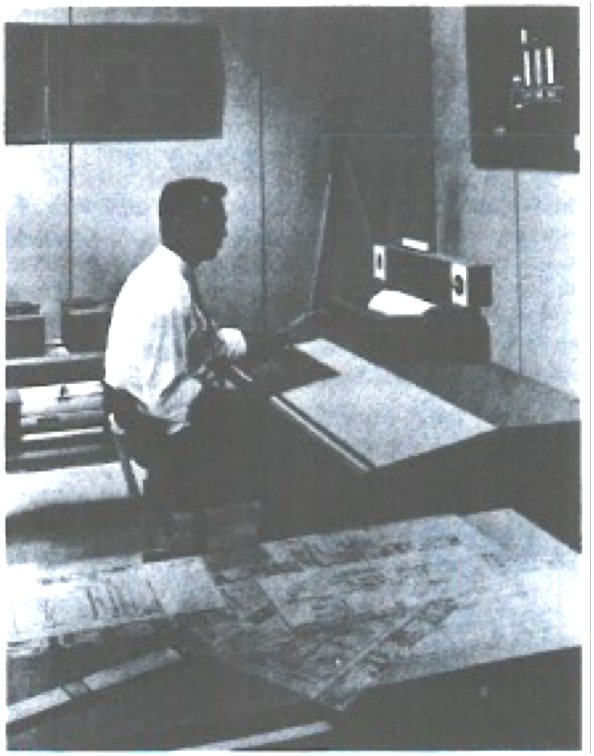
Chaque type d'automate est appelé à « connaître » d'un certain type de *langage* (au sens qui sera précisé plus loin), c'est-à-dire qu'il peut reconnaître un tel langage, ou l'engendrer (ou les deux).

Dans l'analyse des langues naturelles les automates apparaissent aujourd'hui comme pouvant fournir des modèles, ce qui permet de justifier l'appellation de « langage » que l'on donne aux langages de programmation.



Si l'ordinateur frappe l'imagination des foules, il le doit aussi à son aspect extérieur. La communication passe par l'intermédiaire d'un « tableau de bord » où les scintillements de lampes-témoins accusent encore -- comme dans une boutique d'horoscopes de foire -- le côté magique du dialogue entre un homme et des électrons. La complexité du tableau de cet ordinateur de la troisième génération, qui possède une mémoire de 524 288 chiffres décimaux, n'est qu'une image de la difficulté d'ajuster les comportements de la machine et de son opérateur : les tableaux et les voyants de contrôle en occupant une partie. (Cliché IBM).

Une machine reçoit un mot de son interlocuteur. Son rôle est de réagir, suivant certaines règles préétablies, à l'ordre qu'il contient et de répondre par un autre mot : ainsi schématisé, un ordinateur est un type particulier d'automate.



L'ordinateur (ici un IBM 1130) est-il, comme le suggère cette photographie, la règle à calcul du mathématicien et de l'ingénier moderne ? S'il est peut-être hardi de le prétendre aujourd'hui, le faible encombrement de cet appareil et la simplicité du langage de programmation utilisé (très proche de l'écriture mathématique) laissent penser qu'il n'en sera plus de même dans un avenir proche. Il est également possible que l'extension de la consultation d'un ordinateur par téléphone ou par télétype détourne les utilisateurs potentiels de la solution coûteuse d'une machine permanente à domicile. (Cliché IBM).

Les langages et leur usage

On appelle ici *langage* tout système organisé de signes propre à assurer la communication entre l'homme et la machine.

Les langages dont nous traiterons ne concerneront que les calculatrices digitales (encore appelées numériques) pour lesquelles les messages d'entrée et de sortie sont constitués de suites *discrètes* de signaux élémentaires. L'ensemble des types de signaux d'entrée est fini ; nous désignerons par $X = \{ x_1, x_2, \ldots, x_m \}$ cet ensemble qu'il sera commode de nommer un *alphabet*, les éléments en étant des *lettres* (ou symboles). A titre d'exemple, disons que le langage Algol est tout entier basé sur 116 de ces signes (26 majuscules, 26 minuscules, 10 chiffres, certains signes tels que + - x / ÷ ↑ = < > ≤ ≥ ≠ ( ) [ ] , . : ; : = ∧ ∨ ≡ ⊃ ¬ " ␣ , ainsi que 24 symboles complexes tels que *début, fin, vrai, jusqu'à, commentaire, booléen*, etc.) [(1)] : *début* doit, par exemple, être considéré comme un tout, bien qu'il soit visiblement formé de parties distinctes -- ce qui n'est pas sans troubler … le débutant.

(1) Déplorons à cette occasion les mutilations que doit subir l'Algol faute de machines à perforer complètes.

Une séquence rangée de lettres (ou, plus exactement, une séquence rangée d'occurrences de lettres), est un *mot*. Ainsi LAMBDA, ou L␣équation␣a␣2␣ racines␣réelles sont des mots. Un mot immédiatement suivi d'un autre mot forme par (concaténation) un nouveau mot. Cette opération n'est pas commutative, puisque concaténer le mot *mis* au mot *a* donne, suivant le cas : *misa* ou *amis*.

Elle est néanmoins associative : on peut d'abord concaténer ␣ministre à premier, puis le tout à vice-, ou concaténer vice- et premier avant d'ajouter ␣ministre. Le résultat sera toujours le mot vice-premier ␣ ministre.

Avec le mot « vide », qui ne modifie pas les mots auxquels on le relie, l'ensemble des mots que l'on peut écrire à l'aide des lettres de l'alphabet *X* est un demi-groupe à l'élément neutre, ou monoïde (c'est la structure de l'ensemble des entiers positifs ou nuls, muni de l'addition : cette opération est d'ailleurs commutative, à la différence de la concaténation).

On notera *X** le monoïde des mots écrits à l'aide de l'alphabet *X*. Ce monoïde est libre (au sens précis que les mathématiciens donnent à ce terme). Parmi tous les mots possibles, seuls certains seront retenus comme corrects, en fonction de critères soigneusement définis : par exemple, on n'oubliera jamais de fermer une parenthèse ouverte, etc. Règles et signes déterminent un langage formel *L* inclus dans *X**. Une machine, recevant certains mots de *X**, les examine afin de reconnaître s'ils appartiennent bien à *L*. Après confirmation, elle leur fait correspondre d'autres mots du langage *L'* qu'elle emploie dans la voie de sortie (généralement *L'* et *L* sont identiques). L'essentiel de la communication entre l'homme et la machine réside, pour le premier, en l'écriture des mots *L,* qui déterminent les instructions du calculateur, et la lecture des mots de *L'*.

Pour le mathématicien, une machine à traiter l'information se décrit entièrement en termes de langages et de calcul dans les monoïdes : il est parfois difficile au visiteur d'un centre de calcul de ne jamais oublier, une fois assis à un pupitre (et alors qu'il peut à volonté modifier ses questions suivants les réponses déjà obtenues, les enrichir en les précisant, etc.) que son interlocuteur n'est que quincaille, plus ou moins frottée de software. L'illusion est si complète, dans certain cas, d'une conversation réelle avec un collègue très doué qu'on doit faire un effort pour revenir au schéma précédemment proposé : à un mot bien construit sera répondu un autre mot bien construit, déterminé par des processus certes complexes dans le détail, mais parfaitement automatiques. L'ordinateur n'a rien créé : il a rapidement actualisé une potentialité entièrement déterminée.

Langage machine et langage symbolique

L'ensemble des instructions que la machine accepte constitue un langage. Mais celui-ci est peu puissant, étant avant tout conçu pour commander des mécanismes ; aussi est-il analytique et codé ; ce n'est qu'après avoir subi un certain nombre de transformations qu'il devient vraiment lisible.

```
PRØGRAM PRØDUIT
DIMENSIØN A (50, 50), B (50, 50), C(50, 50)
READ 6,A ; READ 6,B
6   FØRMAT (10F 8.3)
    D ∅ 5 I = 1, 50
    D ∅ 5 J = 1, 50
5   C (I, J) = 0
    D ∅  7 I = 1, 50
    D ∅  7 J = 1, 50
    D ∅  7 K = 1, 50
7   C (I, J) = C (I, J) + A (I, K) * B (K, J)
    D ∅  8 I = 1, 50
8   PRINT 9, = (C (I, J), J = 1.50)
9   FØRMAT (N, (5X,10 (F8.3, 4X))
     END
```

Programme en Fortran effectuant le produit de deux matrices 50 X 50. On remarquera les ordres D ∅ 5 I = 1,50 qui provoquent l'exécution des calculs jusqu'à l'instruction étiquetée 5 répétée cinquante fois en augmentant de 1 la valeur de l'indice l à chaque fois, et l'instruction C (I, J) = C (I, J) + A (I, K) * B (K, J) qui paraît absurde sur le plan mathématique mais doit être comprise comme « ajouter au contenu de la mémoire repérée dans le programme par l'adresse symbolique C (I, J) (variable indiciée) la valeur du produit du contenu des mémoires
A (I, K) * B (K, J) et ranger le résultat à l'adresse symbolique C (I, J) ». A la main, l'exécution de ce calcul demanderait 12500 multiplications de nombres de 7 chiffres.

N.D.LR. : le symbole original **V** a été remplacé par *.

```
Début réel A, B, C, D, X ;
LECTURE DES DONNEES : A :=     DONNEE ;  B :=
DONNEE ; C := DONNEE ;
Si A ≠ 0 alors aller à SECOND DEGRE ;
 si B ≠ 0 alors
 début COMPOSER (-1) ; COMPOSER (B) ; COMPOSER (C)
        COMPOSER (-C/B) ;
        aller à SORTIE fin ;
 COMPOSER ( si C = 0 alors -3 sinon -2) ;
                aller à SORTIE ;
 SECOND DEGRE : D := B ↑ 2 – 4XAXC ;
 si D < 0 alors aller à COMPLEXE ;
 si D = 0 alors
        début COMPOSER (1) ; COMPOSER (A) ;
              COMPOSER (B)
              COMPOSER (C) ; COMPOSER ( - B/ (2XA)) ;
              aller à SORTIE fin ;
 COMPOSER (2) ; COMPOSER (A) ; COMPOSER (B) ;
        COMPOSER (C)  ;
 X := - (B ÷ (si B > 0 alors 1 sinon -1)
            X RAC (D)) / (2XA) ;
 COMPOSER (X) ; COMPOSER (C / (AXX)) ,
            aller à SORTIE ;
 COMPLEXE : COMPOSER (0) ; COMPOSER (A) ;
          COMPOSER (B) ; COMPOSER (C) ;
 COMPOSER (-B/2/A) ; COMPOSER (RAC (-D)/2/A) ;
 SORTIE = LIGNE ; aller à LECTURE DES DONNEES fin.
```

Programme en Algol de résolution d'une équation du second degré. On repère aisément les opérations auxquelles nous sommes habitués. On remarquera la plus grande souplesse du langage, notamment en ce qui concerne l'utilisation des ordres conditionnels et des procédures. Lorsque l'on ne dispose pas de tous les caractères nécessaires (par exemple <) on tournera par une périphrase (par exemple, INF.) acceptée par le compilateur.

Un homme normal, non spécialement entraîné, ne peut lire globalement un message codé que si celui-ci est de l'ordre d'une ou deux lignes. Au-delà, l'effort est trop grand. Des considérations analogues valent pour l'écriture en code (et n'oublions pas qu'il faut pouvoir se relire). Pour de longs messages, on passerait son temps à faire des erreurs et à essayer de les corriger… en introduisant d'autres. Afin d'obvier à ces inconvénients, il faut que les langages employés par les usagers se rapprochent le plus possible de leur moyen ordinaire d'expression (disons, par exemple, anglais + mathématiques ou français + comptabilité, etc). C'est ce souci qui a conduit aux langages algorithmiques évolués dont les plus connus du grand public sont évolués dont les plus connus du grand public sont Fortran, Algol et Cobol. L'usager écrit donc dans un langage qui ne lui impose pas d'insupportables contraintes. Bien entendu, la machine ne le « comprend » pas, mais grâce à un programme spécial, dit de compilation, elle traduit ce langage évolué et synthétique en un langage analytique. Par la suite, dans une seconde phase, elle exécute le programme dont elle vient d'élaborer la traduction.

Fortran est le plus ancien des langages évolués, le plus répandu peut-être (dans ses versions successives) grâce à la compagnie industrielle qui l'a mis au point et qui contrôle aujourd'hui la plus grande partie du marché.

Algol, né en 1958, se rapproche beaucoup plus que Fortran du langage mathématique « ordinaire ». Les scientifiques qui ont créé Algol ont eu le souci d'assoir leur construction sur une base logique. Pour la première fois, on a vu un langage décrit par une grammaire entièrement explicitée et formalisée. En Algol, la notion informatique de procédure correspond parfaitement à la notion mathématique de *fonction*.

Tandis que Fortran et Algol visent surtout à recouvrir les besoins de l'analyse numérique, Cobol a été conçu en vue de la gestion. Il n'a pas connu fortune éclatante.

Les mots que comprend la machine appartiennent à des langages peu compréhensibles par l'homme.
Il faut transiter, à l'entrée comme à la sortie, par des langages plus proches des mathématiques et de la langue vulgaire.

On compte aujourd'hui des milliers de langages (sans parler des variétés dialectales). Les raisons de ce babélisme ne sont pas toujours avouables. Pour rester positifs, disons que chaque domaine spécifique appelle un langage spécifique. Nous avons signalé l'analyse numérique, la gestion ; mais il y a aussi la physique des hautes énergies, avec ses chambres à traces et ses clichés à dépouiller ; il y a les applications linguistiques, et bien d'autres.

Le temps partagé permet à chacun de croire n'utiliser la machine que pour son usage personnel alors qu'elle met à profit la faiblesse humaine (temps de réponse très longs, hésitation devant un ordre à passer, etc.) pour se prêter successivement à chacun sans jamais être donné toute entière. Mais si de expérimentations très développées ont déjà eu lieu en grand nombre (ici, à l'institut Blaise-Pascal), l'adaptation effective du matériel existant à cette technique pose encore de délicats problèmes. (Cliché IBM.)

```
procédure algorithmique d'Euclide (a, b) résultats :
 (pgdc) ;
 commentaire calcul du pgdc de a et de b, lequel
  est égal à 0, par convention, si a x b = 0 ;
 entier a, b, pgdc ;
 début
      entier dividende, diviseur, reste ;
      si (a = 0 ou b = 0) alors
           début pgdc : = 0 ; aller à e1 fin
                sinon
      diviseur : = a ; reste := b ;
      pour dividende := diviseur tant que reste ≠ 0
                        faire
          début diviseur := reste ;
                reste := dividende – (dividende *
                          diviseur) x diviseur
           fin ;
      pgdc := diviseur ;
e1 :
fin
```

Calcul du plus grand diviseur commun de deux nombres.

L'idée se fait jour de créer des langages universels, formés en quelque sorte de pièces détachées standard, avec lesquelles on pourrait, à la demande, fabriquer (à l'aide de procédures) les êtres et les opérations de tel ou tel langage spécifique et spécificité. Un exemple d'un tel langage est l'ATF (langage « à-tout-faire » mis au point à l'Institut Blaise Pascal. Les concepts théoriques sur lesquels il se fonde sont très abstraits : ils viennent de la théorie des automates et de la logique combinatoire. Les applications en sont très concrètes : c'est ainsi que l'ATF de gestion, conçu entre autres pour manipuler rationnellement les fichiers (le mathématicien dirait : les suites d'arborescences), semble rencontrer la faveur des gestionnaires.

Il y a un immense intérêt à écrire les programmes dans des langages évolués. Un premier avantage est qu'on les comprend : ils assurent la communication homme-homme. Mais surtout, quand on change de machine, on n'a pas à réécrire toute une bibliothèque de programmes : il suffit de réécrire le compilateur.

Du point de vue de l'usager, le bénéfice est non moins considérable. Au prix d'un apprentissage relativement aisé, il est en mesure de se servir lui-même. Naguère, il devait avoir recours aux services programmeur. Cette étape était une source de perte de temps et de conflits entre le client, peu rompu aux exigences très spécial du calcul automatique souvent incapable de décrire avec précision sa méthode ou son problème à un profane, et le programmeur, dont la culture personnelle est forcément limitée, qui n'arrive pas à obtenir de son interlocuteur les renseignements absolument indispensables à une écriture correcte. Cette collaboration paralytique / aveugle doit faire place à une intervention directe du demandeur au pupitre. Au bout d'un mois ou deux de travail, le client (géologue ou banquier) doit être capable d'écrire lui-même ses instructions, de les améliorer empiriquement selon les heurs et malheurs de son calcul, et d'en extraire les réponses

cherchées. Si l'aide d'un homme de métier reste nécessaire, pour des raisons évidentes, celui-ci est passé du rang de chaînon indispensable à celui de conseiller technique, le maitre d'œuvre étant la personne la plus intéressée à la bonne marche du travail et la plus capable de le modifier selon les évènements.

Il faut ajouter que, jusqu'à une date récente, des exigences tout à fait normales de rentabilité interdisaient d'offrir la disposition de la machine à des amateurs, même éclairés. Avec la technique du temps partagé (time-sharing) les choses sont sur le point de changer.

Plusieurs personnes, assises à des pupitres indépendants, écrivent des mots indépendants de façon aléatoire. La calculatrice doit mettre à profit le temps pendant lequel M. X. écrit au clavier pour calculer au profit de M. Y. Entre la réception d'un programme et son exécution, la traduction des mots conventionnels en impulsions électroniques est donc compliquée d'une procédure de choix des priorités que l'on commence à maîtriser maintenant, bien que peu de matériels aient réellement été conçus dans cette optique, imposée aux constructeurs par les usagers (et non l'inverse, comme on aurait pu croire).

Perspectives pour l'utilisateur

Le temps approche où les personnes qui ont besoin des services d'une machine auront la possibilité de communiquer directement avec elle en employant un langage structuré selon l'entendement humain.

Bien que construites en fonction d'une logique proche de celle de l'homme, les machines s'en distinguent parce qu'elles peuvent traiter comme calcul. Leur puissance est due à leur rapidité, à l'étendue de leur mémoire et à ce que, grâce aux ruptures de séquence, elles peuvent aller chercher des séquences de calcul récurrentes chaque fois qu'elles en ont besoin. Actuellement, elles sont incapables de lire des caractères imprimés non préalablement standardisés. Ce problème doit toutefois être bientôt résolu et elles resteront encore longtemps très limitées quant à leur possibilité d'utiliser des informations non préparées exprès pour elles. Par contre, elles effectuent en quelques minutes des calculs qui demanderaient sans elles plusieurs mois.

N'importe quel utilisateur peut se familiariser très rapidement avec les langages de programmation dits « évolués ». Que ceux-ci aient des propriétés mathématiques intéressantes n'a pas plus d'intérêt pour lui que la grammaire d'une langue naturelle n'en a pour un enfant de deux ans qui apprend à parler.

Par contre, le programmeur qui connaît déjà sa langue de base gagnera à s'interroger sur la structure de ce qu'il écrit. Et, surtout, une amélioration de la théorie des langages formels et de l'axiomatique des automates finis ne peut que conduire à des progrès dans les possibilités offertes par des langages plus rationnels. La construction des machines elle aussi pourrait y gagner. A voir la façon anarchique dont se sont multipliés les exemplaires de ces coûteux hochets qui pourrissent doucement, avec un taux d'utilisation de l'ordre de 3%, on peut croire que des études de langages et de grammaires auraient pu faire économiser quelques millions à une collectivité qui ne sait pas encore très bien quel comportement adopter vis-à-vis de ces étranges machines.

… et perspectives pour l'utilisation

Les machines apportent à l'homme une aide irremplaçable en raison, uniquement, de leur puissance : elles permettent de traiter en quelques minutes des problèmes qui demanderaient des mois de travail manuel, et en quelques heures des problèmes qu'on n'eût pas même songé à poser avant leur apparition.

De nombreuses variantes sont nécessaires pour adapter les « mémoires » des ordinateurs aux tâches particulières de chacun. Entre les exigences contradictoires de grande capacité et de vitesse d'interrogation, il faut trouver des solutions qui tiennent compte de chaque situation (on voit ici une mémoire à cellules magnétiques, divisée en sous-cellules et en feuillets enfermés dans un étui facilement maniable). L'apparition de mémoires réellement riches a modifié du tout au tout (comme l'adoption des langages de programmation) les conditions d'utilisation des machines : les premiers exemplaires historiques en étaient dépourvus, ce qui rendait très complexe leur utilisation.
(Clichés IBM.)

En revanche, elles imposent des contraintes insupportables qui sont essentiellement de deux ordres : tout d'abord, elles ne « comprennent » vraiment que des langages étranges pour l'entendement humain. Ensuite, en raison de leur coût, elles sont de merveilleux prétextes à la mainmise sur leur gestion d'administrateurs en mal de pouvoir … ou en quête d'occupation. Et quand on se défend d'être un technicien, quand on se méfie en outre des hardiesses de l'imagination, on est conduit tout naturellement à imiter des solutions éprouvées… il y a un demi-siècle : on crée un poste de télégraphe par canton ou arrondissement, on prépose un agent à la traduction des messages, on érige un guichet : les clients devraient être sensibles à l'honneur qui leur est fait d'y venir faire la queue.

Fort heureusement, quelques machines ont réussi, jusqu'ici, à échapper à cette emprise. Ceux qui les emploient ont songé avant tout à épargner leur peine et celle de leurs semblables. Ils ont créé et créent encore des langages de programmation directement compréhensibles à tout esprit normalement constitué, et indirectement aux machines grâce à des programmes de traduction. Ils ont songé aussi à épargner leurs pas et à éviter l'attente irritante et improductive en commandant la machine à distance. Il leur reste à parfaire les programmes qui répartiront, dans la machine, le temps et la place entre les divers usagers.

Ainsi, en se souciant avant tout, de l'aspect humain du problème de la communication homme-machine, ils sont sur le point de parvenir à une solution plus efficace et plus rentable que la solution « administrative » : le télégraphe de M. Chappe n'a pas changé, mais on peut déjà rédiger soi-même ses messages ; et bientôt, espérons-le, on pourra même les téléphoner.