ANNALES SCIENTIFIQUES
DE L'UNIVERSITÉ DE CLERMONT-FERRAND 2
Série Mathématiques

YVES HARRAND

Programmation automatique simplifiée

Annales scientifiques de l'Université de Clermont-Ferrand 2, tome 8, série *Mathématiques*, n° 2 (1962), p. 129-130

<http://www.numdam.org/item?id=ASCFM_1962__8_2_129_0>

PROGRAMMATION AUTOMATIQUE SIMPLIFIÉE

Yves HARRAND
Ancien élève de l'École Polytechnique
Attaché à la direction de la Société Européenne pour le Traitement de l'Information

Pourquoi le langage des machines à calculer est-il encore aussi éloigné du langage humain de l'analyse numérique, après 15 ans de développement ? Il semble que la réponse soit double : d'abord, ces machines ont été étudiées pour répondre à des besoins de plus en plus variés, comme la gestion des entreprises, la commande des processus industriels, la traduction automatique des langues, la documentation automatique, etc. Un langage capable de répondre à des besoins aussi vastes, devait devenir très analytique, si l'on voulait éviter que le "vocabulaire", c'est-à-dire les types d'instructions, soit très riche : pour éviter une telle solution, toujours onéreuse, les constructeurs ont été conduits à un langage constitué d'éléments très courts, se rapprochant plus des langues syllabiques de l'Orient que des langues occidentales !

Ensuite, la rapidité de plus en plus grande des mémoires a incité à recourir à la microprogrammation, pour remplacer des fonctions complexes par des sous-programmes. Au contraire, le langage mathématique évolue vers une forme de plus en plus synthétique ; il apparait donc nécessaire, dans l'impossibilité de combler immédiatement le fossé qui s'est creusé entre langage humain et langage des machines, de jeter au moins un pont. Ce pont sera constitué par les traducteurs automatiques, réalisant la programmation automatique.

Avant de décrire les diverses solutions qui peuvent être envisagées, il est bon de noter qu'il faut au préalable réduire le langage humain à un certain formalisme : malgré les apparences, le langage mathématique n'est pas toujours dépourvu de redondance ou pire, d'ambiguité ! Les opérations portant sur des variables indicées en sont un exemple, puisque l'on trouve :

$$a_{i,j}\, b_{j,k} \qquad a_i^j\, b_j^k \quad \text{etc.}$$

L'emploi de petites machines, à mémoires limitées, impose certaines contraintes, soit sur la méthode de traduction, soit sur le langage formel dans lequel le programme sera rédigé. On peut songer à réduire le "pont" en traduisant en une langue plus synthétique que le langage machine ; mais le programme ainsi obtenu devra être interprété à chaque exécution ; cette dernière solution est chère en mémoire, le programme-interpréteur devant être présent à chaque exécution, et en temps, chaque instruction en code symbolique intermédiaire devant être traduite à chaque fois qu'elle est rencontrée dans le programme.

Il nous est donc resté, comme solution permettant de réduire l'intervalle entre les deux langages, de simplifier le langage formel du "programme"objet". Néanmoins, il était utile de tenir compte de l'expérience des réalisateurs de tels langages : il en est un en particulier qui a été réalisé avec l'intention d'en faire un langage universel : l'ALGOL. Sans vouloir le décrire, rappelons brièvement sa structure.

L'ALGOL comporte deux types de phrases, permettant de décrire le programme à traduire en langage-machine : les instructions et les déclarations. Les premières expriment les différentes phases du processus de calcul, les secondes précisent au préalable les caractéristiques de certaines variables : variables entières ou booléennes, dimensions des vecteurs et des tableaux ou matrices, etc. Les instructions sont séparées en plusieurs parties distinctes par des "délimiteurs", qui sont soit des signes de ponctuation, soit des mots du langage usuel (anglais). Enfin, il faut noter la possibilité de grouper une suite d'instructions et de déclarations sous un vocable unique (une étiquette) par l'organisation en "blocs" ou en "procédures".

Malgré ses éminentes qualités, l'ALGOL présente un certain nombre de défauts du point de vue de la réalisation d'un traducteur utilisable sur une petite machine : sa très grande généralité a conduit à une redondance un peu excessive, et l'emploi de trop de délimiteurs "verbaux" comme

"ELSE, THEN, WHILE, etc." alourdit aussi bien la rédaction du programme-objet que celle du programme traducteur qui a la charge de les reconnaitre. Nous avons donc décidé de rester fidèle à l'esprit de l'ALGOL mais de ne pas en respecter la lettre, et d'autre part de compléter la liste des instructions ou des délimiteurs pour faciliter certaines opérations, comme les entrées-sorties ou la mise au point du programme. C'est ainsi que des instructions LIRE et DEMANDER ont été introduites, permettant dans le dernier cas de faire répéter par la machine le nom de la variable dont la valeur doit être introduite. Un signe de "traçage" provoquera, selon la position d'un commutateur, l'impression ou non de valeurs intermédiaires au cours du calcul. Mais afin de mieux faire sentir les points où l'effort de simplification a porté, nous allons décrire sommairement ce nouveau langage, le MAGE, en établissant un parallèle avec l'ALGOL.

En vue d'utiliser au mieux la mémoire de la machine, le traducteur MAGE effectue son travail au fur et à mesure de la lecture du programme-objet, et évacue périodiquement sous forme de ruban perforé ou de bloc sur bande magnétique, le programme déjà traduit. Le programme en langage-machine se constitue donc toujours dans la même petite zône de mémoire (256 mots). Les variables sont désignées par des groupes quelconques de lettres ou chiffres, mais au contraire de l'ALGOL seuls les 3 premiers caractères sont pris en considération. Ceci allie la clarté du texte en clair, où peut figurer par exemple : ALPHA, à la simplicité de constitution du "dictionnaire".

La simplification des instructions sera illustrée par l'exemple de l'instruction d'itération ; à l'instruction ALGOL :

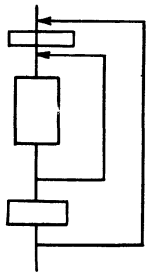
FOR I : = 1 STEP 2 UNTIL 30 DO :

correspond en MAGE :

POUR : I = 1,2,30 (retour chariot)

A ce propos, signalons que le bloc ALGOL n'a son équivalent en MAGE que pour les instructions d'itérations, car dans les autres cas on peut s'en passer, grâce à l'emploi des Procédures qui existent en MAGE. Ainsi un ensemble de 2 boucles imbriquées s'écrira :

```
POUR : I = 1,2,30
.....
POUR : J = 1,1,45
.....
.....
RETOUR :
.....
.....
RETOUR :
```



Il n'y a aucune ambiguité dans les boucles, une boucle ne pouvant être que totalement intérieure ou extérieure à une autre boucle.

Dans le cas des procédures, la liste des paramètres a été supprimée, en laissant le soin au programmeur de définir les équivalences par une nouvelle instruction : POSER :

Notons qu'une certaine souplesse a été introduite dans l'emploi des procédures ; celles-ci peuvent être placées n'importe où dans le programme, la fin de la procédure étant signalée par le "délimiteur" : REPRISE.

Il parait inutile de décrire plus complètement le MAGE ; ce qui vient d'être dit suffit pour montrer par quelles voies on s'est efforcé d'adapter à une petite machine, dont la capacité de mémoire peut n'être que de 2 300 mots, le style des langages primitivement utilisés sur des machines beaucoup plus puissantes.