D | A propos d'un langage universel

Par J. Poyen, Collège de France, C.N.R.S. (France)
et B. Vauquois, Faculté des Sciences, Grenoble

Les auteurs envisagent un langage universel du point de vue de l'utilisateur scientifique dont le seul souci est l'exécution rapide de son problème, abstraction faite des sujétions technologiques propres aux différents calculateurs. Dans cette optique, un tel langage doit être d'un apprentissage rapide et facile, doit permettre la formulation de tout problème scientifique et doit autoriser l'emploi de toute machine suffisamment puissante.
Tout d'abord, on analyse rapidement les résultats obtenus avec les systèmes de programmation automatique existants. Malgré les progrès réalisés, ces systèmes ne permettent pas de se libérer complètement des servitudes de programmation des calculateurs.
On est ensuite conduit à examiner les conditions que doit satisfaire tout projet de langage universel pour atteindre le but proposé à l'introduction; on insistera sur les caractères de clarté d'écriture, de souplesse d'emploi et de facilité d'apprentissage. A la lumière des critères ainsi définis, on analyse le langage algorithmique proposé par la Conférence de Zurich (27 mai à 2 juin 1958). Le présent projet s'appuie sur la classification des êtres mathématiques, adoptée à cette conférence. S'il conserve certains aspects du langage algorithmique, il présente cependant des divergences de forme quant à l'écriture des éléments de programme et propose un nouveau système de rédaction.
Le langage ainsi défini est adapté aux seuls besoins de l'utilisateur scientifique et ne saurait être limitatif. Sa structure lui permettrait d'être inclus dans un langage «élargi» destiné à la rédaction de l'ensemble des problèmes traitant l'information.

Suggestions for a universal language. This paper considers a universal language designed for use by scientists whose sole concern is to solve their problems rapidly, without regard to the technical difficulties in coding for different types of computer. Such a language must therefore be quick and easy to learn, suitable for the formulation of all scientific problems, and usable with any sufficiently powerful machine. It begins by making a rapid analysis of the results obtained with existing systems of automatic programming—none of which, despite the progress made, have yet escaped from programming conventions imposed by the computer.
It then discusses the conditions which any proposed universal language must fulfil in order to serve this purpose; in particular, clarity of script, flexibility of use and ease of learning are all essential. In the light of these criteria, the authors examine the algorithmic language proposed at the Zurich conference (27 May to 2 June 1958). The universal language now proposed is based on the classification of mathematical entities which was adopted at that conference. Although it retains certain features of the algorithmic system, it embodies some differences in the forms used for transcribing the programme components, and proposes a new method of drafting.
This language is designed to meet the needs of scientists only, and makes no claim to be exhaustive; but its structure is such that it could be incorporated in an "expanded" language suitable for the drafting of all matter in the field of information processing.

Bemerkungen über eine Universalsprache. Die Autoren betrachten eine Universalsprache aus der Sicht eines wissenschaftlichen Benutzers, dessen einziges Bestreben darin besteht, seine Aufgabe schnell durchzuführen. Von den Besonderheiten der Programmierung verschiedener Rechenautomaten wird abgesehen. Unter diesem Gesichtspunkt muß eine solche Sprache schnell und leicht erlernbar sein, die Formulierung jedes wissenschaftlichen Problems gestatten und den Gebrauch jeder genügend leistungsfähigen Maschine ermöglichen.
Zuerst werden in Kürze die Ergebnisse besprochen, die man mit vorhandenen Systemen zum automatischen Programmieren gewonnen hat. Trotz der erzielten Fortschritte erlauben diese Systeme nicht, sich vollständig von einer Bindung an das Programmieren von Rechenautomaten zu befreien. So wird man dazu geführt, die Bedingungen zu untersuchen, denen jeder Entwurf einer Universalsprache genügen muß, um das eingangs vorgeschlagene Ziel zu erreichen. Man muß folgende Charakteristika fordern: Klarheit der Schreibweise, Geschmeidigkeit im Gebrauch, leichte Erlernbarkeit. Im Lichte dieser Kriterien wird die algorithmische Sprache untersucht, die auf der Züricher Zusammenkunft (27. Mai bis 2. Juni 1958) vorgeschlagen wurde. Der jetzige Entwurf stützt sich auf die Einteilung der mathematischen Gegebenheiten, die auf der Zusammenkunft angenommen wurde. Zwar behält der Entwurf gewisse Gesichtspunkte der algorithmischen Sprache bei, zeigt aber Abweichungen in der Schreibweise der Programmelemente und schlägt ein neues Abfassungssystem vor.
Die so definierte Sprache ist einzig den Aufgaben eines wissenschaftlichen Benutzers angepaßt und dürfte noch erweiterungsfähig sein. Ihr Aufbau würde erlauben, sie in eine „erweiterte" Sprache einzugliedern, die zur Abfassung der Gesamtheit von Problemen der Informationsverarbeitung bestimmt ist.

К вопросу об универсальном языке. Авторы рассматривают универсальный язык, предназначенный для научных работников, единственной заботой которых является быстрое выполнение их задачи, причем игнорируются технологические трудности в кодировании свойственные различным вычислительным машинам. В свете этого, изучение такого языка должно проходить легко и быстро, он должен позволять сформулировать любую научную задачу, а также допускать использование любой достаточно мощной машины. Сначала кратко анализируются результаты, полученные с существующими системами автоматического программирования, ни одна из которых, несмотря на достигнутые успехи, эти системы не позволила все же полностью освободиться от услуг программистов.
Затем рассматриваются условия, которым должен удовлетворять любой проект универсального языка, для того чтобы была достигнута эта цель, подчеркивается ясность записи, гибкость применения и легкость обучения. В свете определенных таким образом критериев производится анализ алгорифмического языка, предложенного на конференции в Цюрихе (27 мая — 2 июня 1958 г.). Предложенный теперь универсальный язык основан на классификации математических элементов, принятых на этой конференции. Несмотря на то, что он сохраняет некоторые аспекты алгорифмического языка, он все же дает отклонения от формы в отношении записи элементов программы, и предлагает новую систему редактирования.
Определенный таким образом язык может быть использован только научным работником и не должен иметь ограничений. Его структура позволяет включить его в «расширенный» язык, предназначенный для редактирования ряда задач, относящихся к обработке информации.

Sobre un lenguaje universal. Los autores consideran un lenguaje universal desde el punto de vista del científico que ha de utilizarlo, cuya única preocupación se cifra en la rápida solución del problema que tiene entre manos, independientemente de las exigencias técnicas propias de los distintos tipos de calculadoras. Las condiciones que debe satisfacer un lenguaje así concebido son la sencillez y rapidez de su aprendizaje, la posibilidad de formular con su ayuda todo problema científico y, finalmente, su adecuación a toda máquina suficientemente potente.
Se efectúa, ante todo, un análisis de los resultados conseguidos mediante los sistemas de programación automática que existen en la actualidad. Aún teniendo en cuenta los progresos llevados a cabo, tales sistemas no permiten independizarse del todo de las servidumbres carcterísticas de la programación de las calculadoras.
Se estudian a continuación las condiciones que ha de satisfacer todo proyecto de lenguaje universal que pretenda alcanzar dicho objetivo; los rasgos principalmente considerados serán la claridad de escritura, la flexibilidad y agilidad de su manejo y la sencillez de su aprendizaje. Tomando como punto de referencia los criterios así definidos, se analiza el lenguaje algorítmico propuesto en la Conferencia de Zurich (27 de mayo — 2 de junio de 1958). El presente proyecto se apoya en la clasificación de los entes matemáticos que se adoptó en la citada conferencia. Aunque conserva algunos de los rasgos del lenguaje algorítmico, el proyecto presenta, sin embargo, ciertas discrepancias formales en lo tocante al modo de escribir los elementos que forman parte del programa, proponiendo, al mismo tiempo, un nuevo sistema de redacción.
El lenguaje descrito se adapta exclusivamente a las exigencias del científico que ha de valerse del mismo y no tiene un carácter limitativo. Su estructura es tal que le permite quedar incluído dentro de un lenguaje "ampliado" destinado al planteamiento de todo el conjunto de problemas relativos a la información.

1. Synthèse des programmations automatiques

Avec l'accroissement de la puissance des calculateurs électroniques, ces dernières années ont vu aussi se développer les méthodes de programmation automatique. En effet, les machines actuelles étant dotées d'un nombre de mémoires quasi illimité (en tout cas, assez grand pour traiter tout problème à l'échelle humaine), elles posent au programmeur des problèmes d'implantation extrêmement ardus; en outre, ces mêmes machines étant capables d'effectuer les opérations élémentaires à très grande vitesse, elles engagent l'utilisateur à aborder des problèmes de plus en plus compliqués: en conséquence, l'étape de la programmation rendue difficile par des causes multiples, risquait de compromettre l'efficacité et surtout la commodité des calculateurs modernes à grande puissance. Aussi, parallèlement a l'effort porté sur la construction des machines, s'est imposé un effort de «super-programmaton» visant à rendre particulièrement facile le problème des communications entre l'homme et la machine. Il ne peut être question d'aborder ici toute l'histoire et l'évolution des diverses techniques qui se sont proposé de rendre assimilables par la machine les programmes écrits en des langages de plus en plus rapprochés du langage humain. Cependant, nous mettrons l'accent sur les dernières réalisations dans ce domaine car leur examen nous conduira aux idées plus générales qui font en grande partie l'objet de cette communication.
Dans cette optique, notre base de départ est constituée par l'ensemble des programmations automatiques dites de «type algébrique» telles que FORTRAN MATH-MATIC, AP 3 . . . Bien que présentant quelques divergences de forme (ces divergences nous apparaissent d'ailleurs comme très légères), ces différents systèmes ont des caractères communs très nombreux qui correspondent à une même manière d'envisager la programmation automatique.
Notre première tâche va consister à préciser cette méthode de pensée devant le problème posé. En l'occurence, l'énoncé du problème est fort simple: Etant donné une machine dont on connaît le code d'instructions avec toutes ses possibilités (et aussi toutes ses finesses), il s'agit de trouver un procédé de programmation qui n'utilise pas ce code, jugé trop analytique, mais qui utilise une formulation aussi voisine que possible de la formulation habituelle des mathématiciens, et qui débarrasse l'utilisateur du plus grand nombre possible des servitudes propres à cette machine. Pour traiter le problème ainsi posé, il faut préciser ce qu'on entend:

a) par procédé de programmation,
b) par formulation aussi voisine que possible de la formulation mathématique,
c) par élimination du plus grand nombre possible de servitudes propres aux machines.

Constatons simplement que les diverses programmations automatiques donnent pratiquement la même réponse à ces trois questions. En effet, le procédé de programmation est envisagé comme une séquence de phrases sommaires exprimées en un langage défini à priori par son vocabulaire et sa syntaxe. Pour définir ce langage, on a cherché d'une part à se servir de la formulation mathématique comme d'une référence, d'autre part à inclure toute la souplesse nécessaire à la traduction d'un organigramme.
Aussi, les premiers objectifs sont-ils d'une part l'écriture telle quelle des expressions algébriques et logiques, d'autre part l'écriture commode des aiguillages et même du calcul sur ces aiguillages. Au cours de ce travail de définition, on a été amené à altérer plus ou moins le langage mathématique car le pseudo-code ne s'écrit que sur une ligne et ne peut donc exécuter une copie fidèle d'expressions contenant des indices (voire des indices d'indices) et des exposants; en outre, on a préféré répartir sur plusieurs phrases des expressions mathématiques portant sur des quantités

indexées, de manière à séparer les opérations portant sur les indices des opérations de calcul proprement dites. Si les diverses programmations automatiques proposent des écritures différentes, elles relèvent toutefois du même point de vue.
De même, les différentes solutions adoptées pour réaliser aiguillages et calculs sur ces derniers, permettent toutes de traduire avec des degrés de commodité équivalents les organigrammes les plus complexes.
Enfin, en ce qui concerne la libération des servitudes propres aux machines, les programmations automatiques actuelles sont toutes «symboliques» (l'implantation est entièrement confiée à la machine) et certaines offrent même quelques avantages supplémentaires.

2. Introduction au langage universel

Ainsi, nous voici en présence de systèmes équivalents, dont la codification est confiée à des machines de puissances équivalentes. En somme, chaque constructeur de grosse machine a fait l'effort de créer une programmation automatique qui lui est propre, valable sur sa seule machine et pourtant très voisine de celle des autres. Or, l'utilisateur, pour de multiples raisons, peut avoir à se soucier beaucoup plus de la programmation que de la provenance de la machine et ressent alors le besoin d'unification de ces systèmes juxtaposés. Citons l'échange de programmes entre utilisateurs qui ne sont pas équipés avec le même matériel; le laboratoire de calcul qui, possédant plusieurs calculateurs différents, perd beaucoup de souplesse. Citons enfin l'utilisateur qui, n'ayant pas de machines, peut bénéficier de durées de calcul limitées auprès de tiers divers sur des calculateurs divers et qui est obligé d'écrire le même programme en autant de langages différents.
Il n'y a que deux issues possibles: ou bien traduire les programmes d'un pseudo-code dans tous les autres (entendre par là une traduction automatique), ou bien disposer d'un pseudo-code unique que toute machine serait capable d'accepter. En fait, ces deux solutions se ressemblent, le premier terme de l'alternative conduisant à une double traduction. Admettons que, de ce point de vue, la solution souhaitable soit celle d'un langage unique et examinons toujours sous cet angle les problèmes qu'elle pose.

1) Un problème de fondement et de logique

Pour ce langage unique, il faudra, tout comme pour les systèmes de programmation, commencer par en définir les éléments: d'abord les symboles élémentaires et leurs divers rôles, ensuite des règles de formation d'agrégats de ces symboles pour obtenir des termes, enfin des règles de construction d'expressions à partir des termes ou à partir d'expressions plus simples. Il semblerait que l'expérience des auto-programmations actuelles puisse accélérer l'étude purement logique du langage en tant que système formel. Mais, à notre connaissance, aucun constructeur n'a fait état d'une telle étude comme préalable à l'élaboration d'un système de programmation automatique; sans doute, a-t-elle eu lieu après une étape guidée par l'intuition pour parfaire le langage définitif. En tout ces, un langage unifié devra fatalement s'y soumettre. Nous reviendrons sur ce point par la suite, et admettons pour le moment que seules l'intuition et surtout l'expérience conduisent au langage unifié; d'autres problèmes, et non des moindres, vont encore se poser.

2) Un problème de degré de complexité

Expliquons ce que nous entendons par là:
D'abord, il faut savoir dans quelles limites le langage unifié sera capable de suivre de près ou de loin la formulation mathématique des problèmes. Ensuite, il faut déterminer quelles sont les posibilités du langage et la concision requise pour traduire le schéma d'un organigramme. Enfin, il importe d'unifier les modes d'entrée des données dans la machine et de sortie des résultats sur les divers organes disponibles à cet effet.
A la lumière de ce qui précède et dans les conditions que nous avons admises, nous sommes naturellement conduits à adopter comme langage unifié un langage équivalent aux systèmes de programmation automatique dont nous avons parlé.

3) Un problème d'intérêt général

En nous plaçant toujours dans l'optique du besoin d'unification, le langage commun n'a d'intérêt que s'il est adopté par tout le monde (sans quoi la double traduction suffit à résoudre les difficultés posée par le nombre de langages existants). A ce sujet, il ne faut surtout pas dissimuler les difficultés quasi insurmontables qui se présentent chaque fois que l'on recherche l'unanimité. Nous serions heureux si l'examen de ces difficultés aboutissait à une entente générale par des suggestions suffisamment souples pour ne pas soulever d'objections majeures.

a) Le vocabulaire de ce langage ne peut pas se contenter des symboles du langage mathématique universel; il est nécessaire d'inclure un certain nombre de mots, empruntés à un langage humain pour tout ce qui concerne les particularités propres à la programmation. Deux solutions sont possibles: ou bien adopter un formalisme complet dont les symboles remplaceraient ce vocabulaire particulier, ou bien adopter une langue vivante admise par tous. Nous verrons plus loin comment la séparation des niveaux d'un langage universel permet de supprimer complètement cette difficulté.
b) La syntaxe de ce langage introduit des difficultés beaucoup plus importantes. En effet, les programmations automatiques existantes ont des syntaxes légèrement différentes bien que se situant au même degré de complexité. Comme les considérations précédentes nous obligent à donner à ce langage unifié ce même degré de complexité, la tentation est grande (et peut-être est-il difficile de faire autrement) de s'approcher de l'auto-programmation particulière à un constructeur. Il est alors indubitable que les autres constructeurs vont réagir de manière très défavorable.

3. Contribution à la construction du langage universel

Il s'agit moins de construire de toutes pièces un nouveau langage (ce qui ne ferait qu'augmenter le nombre des programmations automatiques) que d'analyser un projet déjà admis par certains et d'en déduire un minimum de modifications en vue de surmonter les difficultés que nous avons énumérées.
Le projet en question est celui du langage algorithmique issu de la conférence de Zurich. Avant d'en discuter le vocabulaire et la syntaxe, nous retiendrons l'idée fort intéressante de la séparation des niveaux, à savoir: un niveau de référence, un niveau de publication, un niveau d'utilisation de service («Hardware» représentation). Quelle que soit la forme du langage universel adopté, une telle séparation est la seule méthode qui puisse résoudre les difficultés de vocabulaire; évidemment, la syntaxe du langage devra être la même pour tous les niveaux. En effet, une fois la structure du langage établie, les différents niveaux s'adaptent à la commodité d'emploi dans tous les domaines.
Ainsi, le langage de référence (langage universel proprement dit) pourra être formalisé encore davantage que ne l'est le langage algorithmique. Ce niveau représente, si l'on peut dire, la commune mesure de tous les langages de niveaux inférieurs. C'est aussi à ce niveau que doit se présenter l'étude logique, l'étude des modifications et des extensions que l'on veut apporter au langage. On pourrait d'ailleurs

imaginer que l'existence de ce niveau suffise à donner la solution du problème de la programmation aux utilisateurs. En théorie, cela est vrai car l'utilisateur dispose d'un langage synthétique lui permettant de rédiger un programme avec un nombre restreint d'instructions et le libérant de la plupart des servitudes propres aux machines. En pratique, on cherche encore davantage. En effet, si le langage de référence est très formalisé (et cela est souhaitable), il requiert un apprentissage quelquefois malaisé, il peut être la source d'erreurs d'écriture souvent difficiles à déceler, il devient incompréhensible pour tout lecteur non initié. Si le langage de référence est très redondant, fait appel à un langage humain, il perd ce caractère universel, en ce sens qu'il favorise une langue vivante par rapport aux autres, d'autre part il ne sera plus un système commode pour l'ètude logique.
Donc, pour satisfaire aux exigences de la pratique, le niveau de publication semble le mieux adapté. Conservant la structure fondamentale du langage de référence, ce niveau adoptera un vocabulaire redondant permettant la rédaction d'un programme en un langage très voisin de la language habituelle de l'utilisateur. Il n'y a aucun inconvénient à envisager autant de langages de publications qu'il existe de langues vivantes; en effet, les syntaxes étant identiques, le passage d'un langage de publication à un autre se fait immédiatement par la consultation d'un lexique en machine.

Le langage de service adapte les précédents aux caractères technologiques propres des divers matériels (par exemple, la perforation des lettres sur carte n'obéit pas aux mêmes codes sur les diverses machines, il n'y a pas identité entre les divers signes possibles . . .). Nous ne considérons pas ici ce niveau du langage universel.
Considérant le langage universel au niveau de référence, prenons comme base de travail le langage algorithmique de Zurich. La diversité des ordres et la syntaxe de ce système sont tels qu'ils répondent aux critères des programmations automatiques actuelles. Cependant, il convient de rappeler que toute approche supplémentaire de la formulation mathématique constitue un progrès et que le niveau de référence du langage universel doit être le premier à en tenir compte. A ce sujet, il serait souhaitable d'inclure au langage de référence deux nouvelles opérations:

a) Les opérations Σ et Π du langage mathématique. Suggérons avec les Professeurs Bauer, Rutishauser et Samelson l'inclusion dans le vocabulaire des mots SUM et PRODUCT.

b) Les opérations sur indice temporel. Nous entendons par variable soumise à un indice temporel, une variable dont la valeur évolue non en fonction de la place (indices d'espace appelés communément «indices») mais en fonction du temps. On pourrait alors inclure dans le vocabulaire le mot CYCLE.

En supposant acquises ces nouvelles définitions, nous sommes en mesure de proposer au niveau de publication une écriture satisfaisant à:

— une correspondance bi-univoque (phrase à phrase) avec le niveau de référence,
— un caractère de redondance permettant d'une part un apprentissage facile et rapide, et surtout une lecture immédiatement comprise par toute personne non initiée au langage, d'autre part d'autoriser certaines incorrections qui ne modifient pas la nature du problème (par exemple, fautes de frappe, fautes d'orthographe).

4. Le langage de publication

Le langage se présentera sous forme de phrases, que l'on pourra repérer au moyen d'une étiquette. Chaque phrase a un sens en elle-même et est composée, suivant une syntaxe précise, avec les mots d'un dictionnaire où nous distinguons les mots clés (précisant le sens général de la phrase) et les mots complémentaires (permettant d'assembler d'une façon conventionnelle les constituants de la phrase). Toute phrase comportera au minimum un mot clé.
La communication des programmes se fait dans n'importe quelle langue: un lexique permet la traduction des mots du dictionnaire (une trentaine environ), et la construction des phrases est assez simple pour être commune à toutes les grammaires.

4.1 *Liste des mots clés*

DEFINITION	INDICE
COMMENTAIRE	QUANTITE
CALCULER	OPERATEUR
CONDITION	FONCTION
ALLER A	SI
EXECUTER	FIN
REMPLACER	

4.2 *Les signes de base*

Pour le choix des signes de base, un compromis entre les signes mathématiques usuels, les caractères disponibles d'une machine à écrire et les symboles assimilables par un calculateur est nécessaire. Nous ne retiendrons que les signes indispensables à la formulation d'un problème. Des symboles tels que | | (Module), ! (Fonction factorielle) seront remplacés par des opérateurs (sous programmes).

Nous proposons:

Alphabet latin majuscule: A ··· Z

Chiffres arabes: 0 ··· 9

Signes opératoires arithmétiques: $+ \; - \; \times \; / \; *$ (ce dernier signe désigne l'exponientation)

Signes opératoires logiques: $= \; \neq \; > \; < \; \geq \; \leq$ ET, ou, ′ (apostrophe pour désigner la négation)

Signes de séparation: «Rien»
, pour séparer des quantités de même nature
; pour séparer des parties de phrase de même nature
. pour séparer la partie entière de la partie décimale d'un nombre
( ) pour isoler des groupes de symboles particuliers.

Signes de transfert: = signe qui a pour effet de donner à la quantité désignée à gauche la valeur numérique assignée à la quantité à droite. Remarquons qu'il ne peut y avoir ambiguité avec le signe opératoire logique.

Signe de fin de phrase: dans le langage de publication, se traduira par «Aller à la ligne».

4.3 *Représentation des êtres mathématiques*

a) Les scalaires 1836
18.34501
$2.99 \times 10 * 10$

b) Les opérandes symboliques.

Les opérandes symboliques seront représentés par un caractère alphabétique suivi de caractères alphanumériques, les symboles pouvant désigner un scalaire, une fonction, un opérateur, une variable booleene, une matrice, un nombre complexe, une table, un indice, une variable indicée, une partie d'un programme, un sous-programme, etc.

4.4 *Présentation générale de la phrase*

Les phrases se présenteront comme suit:
1) une étiquette (s'il y a lieu)
2) un mot clé
3) une expression complément du mot clé
4) le signal de fin de phrase (Aller à la ligne).

Exemple

Etiquette mot clé Complément du mot clé

```
8.34   DEFINITION  MATRICES COMPLEXES  A
       B  C;  TABLE  TROL  ENTREE
       ALPHA   MU   SORTIE   Y 1
(  OPERATEUR  FLIP  DEBUT;  ENTREE
   NU  X 7  ZAR;  SORTIE
(  P  Q
(  ..................
(  OPERATEUR  FLIP  FIN
(2 D 2 DEBUT
              ..................
(2 D 2 FIN
```

Les mots clés DEFINITION, OPERATEUR se placent naturellement en debut de phrase.
Un groupe de phrases se repère par les mots DEBUT et FIN précédés de la même étiquette (2 D 2).

4.5 *Ordres de renseignements généraux*

DEFINITION

L'ordre définiton est destiné à préciser la nature mathématique des données du problème, à l'exclusion des scalaires réels (par convention); ces derniers étant les plus fréquents, leur définition alourdirait l'écriture du programme.

Cliché

```
Etiquette  DEFINITION  MATRICE  A BETA  ...;
                       VALEUR COMPLEXE
                       H 1  H 10 ...;
                       ..........
                       SCALAIRE COMPLEXE
                       TIC  TAC  ...
```

COMMENTAIRES

Les commentaires sont destinés à informer le lecteur des caractères généraux ou particuliers du problème. Ils ne font pas partie du programme proprement dit.

Cliché

```
COMMENTAIRE ..........
```

4.6 *Ordres de calculs*

CALCULER

Cet ordre affecte à l'opérande résultat la valeur déterminée par l'algorithme de calcul contenu dans la phrase. Dans ce dernier, ne doivent figurer que des entités de nature mathématique et de valeur numérique connues afin de pouvoir déduire la nature mathématique et la valeur numérique du résultat.

Cliché

```
Etiquette  CACULER  A = ... Algorithme ......
```

CONDITION

Cet ordre affecte à l'opérande résultat la valeur booléenne déterminée par le polynome logique contenu dans la phrase. Les quantités booléennes dans lesquelles interviennent les signes $= \neq > < \geq \leq$ seront mises entre ( ).

Cliché

Etiquette Z = (((A REEL) OU (C $\leq$ 1)' ET (A 1 < × < A 2)) OU (B VRAI) .....)

4.7 *Ordres de saut*

ALLER A

Cet ordre est destiné à modifier la séquence normale des phrases du programme.

Cliché

```
Etiquette  ALLER A  ..........
```

Etiquette de la phrase ou l'on doit sauter.

4.8 *Ordres de commande à distance*

EXECUTER

Cet ordre commande l'exécution d'une phrase, d'un groupe de phrases ou d'un opérateur, répérés par une étiquette (KAPPA).

Cliché

```
Etiquette  EXECUTER  KAPPA
Etiquette  EXECUTER  KAPPA  N  FOIS
Etiquette  EXECUTER  KAPPA  JUSQU'A (C L)
```

C L représente une condition logique telle que (Z vrai
(
(x $\geq$ a

REMPLACER

Cet ordre permet la substitution d'un opérande symbolique, d'une phrase (repérée par une étiquette), d'un groupe dephrases (repéré par une étiquette), d'une fonction ou d'un opérateur, par un être de même nature mathématique.

Cliché

```
Etiquette  REMPLACER  .....  PAR  .....
                      étiquette      étiquette
                      opérande 1     opérande 2
```

Pour redonner au programme son état initial, un nouvel ordre «Remplacer» est nécessaire.

4.9 *Ordres de préparation*

INDICE

Cet ordre précise le domaine de variation d'un indice: aussi bien la suite discrète de ses valeurs que la partie du problème où il doit intervenir. Les indices ne peuvent prendre que des valeurs entières.

Cliché

```
Etiquette  ( INDICE  ALPHA  VARIE DE □ A □
                                    PAR □
           ( INDICE  ALPHA = □ ; □ ; ··· ; □
             ..............
             INDICE  ALPHA  DEBUT
             ..............
             INDICE  ALPHA  FIN
```

ALPHA représente le symbole de l'indice.
□ est un nombre entier (valeur numérique ou symbole défini précédemment).
Le mot DEBUT peut être accolé à la phrase définissant le domaine de variation de l'indice:

Cliché

```
Etiquette  INDICE  ALPHA  VARIE DE □ A □
                          PAR □;  DEBUT
           ......
           INDICE  ALPHA  FIN
```

Si le pas de variation vaut + 1, il n'est pas nécessaire de l'écrire.
Il sera possible d'écrire:

Cliché

```
Etiquette  INDICE  ALPHA = □ ; DEBUT
           ......
           INDICE  ALPHA  PROGRESSE DE □
           INDICE  ALPHA  FIN
```

Les clichés précédents se reportent aux indices d'espaces qui repèrent une quantité dans un ensemble ordonné. Pour signifier qu'un indice est temporel, on emploiera:

Cliché

```
Etiquette  INDICE  TEMPOREL  ALPHA  DEBUT
           ......
           INDICE  ALPHA  FIN
```

Exemple

```
INDICE TEMPOREL T DEBUT
INDICE J VARIE DE 1 A N DEBUT
CALCULER A (J, T) = fonction de A
(J — 1, T) et A (J — 1, T — 1)
SI (A (J, T) — A (J, T — 1) < EPSILON)
ALLER A 1
INDICE J FIN
INDICE T FIN
1 FIN
```

QUANTITE

De même que l'ordre INDICE, cet ordre précise le domaine de variations d'un paramètre. Les clichés sont les mêmes que pour les indices d'espace.

4.10 *Les ordres sous programmes*

OPERATEUR

Cet ordre permet la définition d'un sous programme dont l'exécution sera commandée par un ordre EXECUTER, ordre dans lequel on précisera les valeurs attachées aux paramètres de l'opérateur.

Cliché

```
        Etiquette de l'opérateur
OPERATEUR .....
        Etiquette des paramètres d'entrée
ENTREE ....;....;....;....;....;....;
        Etiquette des paramètres de sortie
SORTIE ....;....;....;....;....;....; DEBUT
.....
        Phrases définissant l'opérateur
.....
        Etiquette de l'opérateur
OPERATEUR ..... FIN
```

Les appellations symboliques des êtres constituant l'opérateur peuvent être réutilisées dans le programme.

FONCTION

Si un opérateur remplit les conditions suivantes:

1) un seul paramètre de sortie,
2) sa définition ne nécessite qu'une seule phrase,

cet opérateur sera appelé «fonction» et le fait d'écrire dans une phrase quelconque l'étiquette de la fonction suivie des valeurs des paramètres d'entrée commande l'exécution de la fonction.

Cliché

```
        Etiquette des paramètres d'entrée
FONCTION FFF (....;...;...;... ...;...;.... ) =
        algorithme de calcul.
```

Dans le langage de publication, on considère comme acquises les définitions des fonctions usuelles (SINUS, valeur absolue, exponentielle, etc....).

4.11 *Ordre conditionnel*

SI

Ce mot conditionne l'exécution de la phrase où il se trouve. Il est toujours accouplé à un autre mot clé.

Cliché

Etiquette SI (Condition logique) { CALCULER ... / ALLER A ... / TRANSFERER ... / EXECUTER ... / REMPLACER ... / FONCTION ... / OPERATEUR ... }
(s'il y a lieu)

Si la condition n'est pas réalisée on saute à la phrase suivante.

4.12 *Ordre d'arrêt*

FIN

indique la fin du problème.

4.13 *Ordres d'entrées sorties*

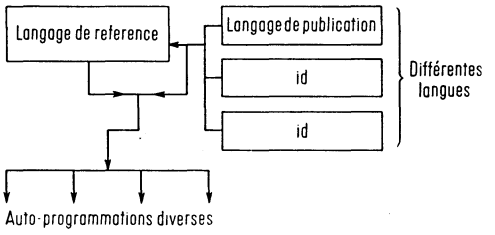
Ces ordres sortent du cadre du langage de référence et du langage de publication. Ils sont liés aux calculateurs et appartiennent au langage de service («Hardware» representation).

5. Conclusion

Dans l'optique de l'unification des langages synthétiques utilisés sur calculateurs, nous proposons, après une analyse du problème, deux solutions:

— au niveau du langage de référence, l'adoption du système algorithmique de Zurich, complété par les définitions de Σ et Π et la notion d'indice temporel;
— au niveau du langage de publication, une écriture en langage humain qui est une image redondante du formalisme de référence.

Ainsi l'on peut imaginer la construction, guidée par le langage de référence, d'un compilateur capable de prendre en charge les programmes rédigés au niveau de publication.



Toujours dans cette optique, le langage reste universel malgré la pluralité des langues. Il semble intéressant de signaler que le niveau de référence est le système bien adapté aux travaux de logique et de définition. Par contre, le niveau de publication est, comme son nom l'indique, mieux adapté à l'écriture et à la communication des programmes entre intéressés.

L'étude actuelle du langage universel s'est proposé de faire la synthèse des programmations automatiques. Mais il apparaît que cette solution d'urgence ne doit pas être considérée comme définitive. Il importe, dans les années à venir, de concevoir un futur langage universel qui serait l'origine et non l'aboutissement des futurs systèmes d'autoprogrammation. Un tel langage ne pourra être valablement établi qu'après une étude complète de la théorie du traitement de l'information, théorie qui reste entièrement à bâtir. Le langage universel sera alors adapté à tous les problèmes passibles d'automatisation, qu'ils soient de nature mathématique ou non.

6. Dictionnaire des mots autorisés dans le langage

A	FONCTION
ALLER A	INDICE
BOOLEEN	JUSQU'A
CALCULER	MATRICE
COMMENTAIRE	OPERATEUR
COMPLEXE	OU
CONDITION	PROGRESSE
DE	QUANTITE
DEBUT	REEL
DEFINITION	REMPLACER
DIMINUE	SCALAIRE
ENTIER	SI
ET	TABLE
EXECUTER	TEMPOREL
FAUX	VARIE
FIN	VECTEUR
FOIS	VRAI

7. Discussion

M. Sangster (Netherlands): The search for a universal programming language which can be fed to many different computers has a complement in the need to improve the communication of analytical mathematics; that is the transfer of analytical deductions to the mind of the reader. Most algebraic texts printed in scientific literature would require smaller steps and much more space to permit the reader to check every step at a glance. The use of a film strip as a form of "living blackboard" might improve the efficiency of the transfer. A more advanced version, however, would be to program every step in logical and universal terms and to use this program as input to a computer equipped with an output devise for displaying the steps in more conventional terms. It might be preferable for didactic reasons to make the input redundant, but it would still be very economical. It might be worth considering this problem together with the requirements for automatic programming so as to produce the common language in its most general form.