UNE APPROCHE LINGUISTIQUE DE LA PROGRAMMATION ?

Jacques ANIS

PRÉAMBULE

Linguiste intéressé par l'informatique, d'abord sous l'angle de l'informatisation de la communication écrite, puis dans la perspective du traitement informatique des langues naturelles, une récente intégration dans l'enseignement supérieur m'a ouvert la possibilité de prendre en charge des unités d'enseignement dans ces deux domaines. Le présent article portera sur l'initiation à la programmation en Pascal que je mène dans le cadre de l'U.E. « Traitement informatique des langues naturelles » (incluse dans la licence de Sciences du Langage de Paris X). Cette U.E. ne compte que deux années d'existence, un an en fait sous sa forme actuelle de 3 heures annuelles (alternance de cours théoriques et de travaux dirigés sur micro-ordinateurs compatibles). Mon but n'est donc évidemment pas d'en faire le bilan mais simplement d'expliciter une démarche(1) qui tire sa spécificité du public visé - il n'est pas encore très fréquent d'enseigner la programmation à des étudiants littéraires - et de l'objectif poursuivi : initier à la problématique de la linguistique informatique. Je ne traiterai ici qu'un seul aspect de mon travail, l'approche du langage de programmation lui-même en tant que système para-linguistique, n'abordant que par le biais des exemples le contenu des petits programmes de traitement de la langue.

On a souvent tendance à assimiler les langages de programmation à des codes au sens restreint du terme ou à des langages formels, comparables par exemple aux formalismes logiques. La pertinence de ces comparaisons varie selon les langages considérés : la première paraît adéquate si l'on envisage les langages dits d'assemblage, dans lesquels

(1) Cette démarche doit beaucoup à Michel Galmiche, maître de conférences de linguistique à Paris III, qui dirige la branche informatique du CARVI de cette université, et à son épouse Marie-France.

sont utilisés des mnémoniques, dans un contexte essentiellement numérique, proche du langage machine ; la seconde semble appropriée, si l'on considère le langage Prolog, issu d'un formalisme logique réduit à des propositions d'un certain type (clauses de Horn). Au-delà de l'hétérogénéité des langages de programmation, la notion de langages de haut niveau donne un certain cadre à la communication homme-machine qu'elle tend à rapprocher de la communication homme-homme : les langages deviennent relativement indépendants des machines, grâce à l'existence de couches intermédiaires ; dans le modèle proposé notamment par Tanembaum(2), on notera qu'un langage de niveau n est assimilé à une machine virtuelle par rapport au langage de niveau n + 1. La traduction du langage de haut niveau en langage machine est assurée soit pas à pas à l'exécution, grâce à des programmes d'interprétation, soit entièrement avant l'exécution, grâce à des programmes de compilation - solution plus rapide, qui a privé les langages d'assemblage d'un de leurs atouts.

Les langages de haut niveau tendent à l'universalité, mais la diversité des applications visées et des milieux concernés ainsi que la concurrence commerciale ont poussé à la création d'un très grand nombre de langages, ainsi qu'à l'éclatement de ceux-ci en dialectes. Cette "babélisation" peut être considérée comme une première ressemblance entre les langages de programmation et les langues naturelles (maintenant abrégés en LP et LN). Le Pascal dit standard coexiste avec d'autres variétés, notamment le Pascal UCSD de l'Université de San Diego et le Turbo-Pascal™ de Borland ; ce dernier a eu un très grand succès commercial et la version 5.5 vient de sortir(3). Assez vite, le Pascal a été implémenté sur micro-ordinateur, pas assez vite cependant pour éviter la diffusion du Basic, lié au départ organiquement à la micro-informatique et ouvert d'emblée aux programmateurs amateurs. Le Pascal a été conçu dès ses origines comme un instrument pédagogique et a connu un grand succès dans les universités américaines pour l'enseignement de la programmation(4). On lui reconnaît des qualités de clarté et de lisibilité : il comporte des aspects déclaratifs, tout en restant fondamentalement

(2) Voir les premières pages de *Architecture de l'ordinateur*, Inter-Éditions, 1987.

(3) Les exemples proposés plus loin ont été écrits dans la version 3. A propos de la version 5.5, il est remarquable que, suivant une tendance générale, y soient introduits des éléments de programmation par objets. En effet, la compétition commerciale crée une tendance anti-Babel : c'est ainsi que quelques années plus tôt étaient sortis les basics structurés où l'on instillait des éléments issus des langages procéduraux.

(4) Ce n'est pas un hasard si Tanenbaum illustre le fonctionnement de l'ordinateur par des procédures écrites en pseudo-pascal.

procédural ; il permet, grâce aux types prédéfinis ou définis par l'utilisateur, de manipuler des données sous une forme structurée ; il est modulaire, de par l'utilisation des fonctions, des procédures et des modules (dans les versions les plus avancées). Ce langage comporte des ressemblances avec d'autres LP (Fortran, Algol, PL/1, C) et l'analyse que nous allons mener pourrait partiellement être transposée à ceux-ci, il nous semble cependant qu'il est tout spécialement comparable aux langues naturelles(5).

LIMITES

Avant de développer la mise en évidence des caractéristiques paralinguistiques du Pascal, il importe, pour éviter tout malentendu, de rappeler le fossé qui le sépare, ainsi que tout LP, des LN : il ne possède évidemment ni l'étendue de leur lexique, ni la variété de leur syntaxe, ni la complexité de leur sémantique ; comme on le rappelle souvent à propos des langages formels, il écarte délibérément l'ambiguïté qui est constitutive des LN ; enfin il n'a pas le large éventail de fonctions sociales de celles-ci. Les textes qu'il permet d'écrire sont des programmes, c'est-à-dire des écrits qui s'évaluent par leur réussite pragmatique. La modalité dominante est l'impératif ; les instances de l'énonciateur et de l'énonciateur sont constamment implicites. Une partie importante des instructions est faite de verbes anglais à forme non marquée interprétés tout naturellement comme des ordres adressés par le programmeur à la machine.

LP & COMMUNICATION

Cependant une dualité fondamentale marque le LP, dans la mesure où s'ajoute à la communication homme-machine - effectuée en réalité indirectement à travers la transposition du texte en code - la communication homme-homme. On sait que l'utilisation systématique des retraits (angl. *indents*) dans la présentation graphique des programmes, que l'on recommande quasi impérativement, n'a aucune pertinence pour la compilation ; elle relève de la lisibilité humaine, permettant à un tiers humain - qui peut d'ailleurs être le programmeur lui-même se relisant - de comprendre plus facilement et plus vite le

(5) C'est sans doute dans les langages orientés objets que l'on trouvera les plus grandes similitudes entre LP et LN (cf. par exemple dans le LP d'*Hypercard Hypertalk* l'emploi du pronom anaphorique *it*).

programme. Cette recherche de lisibilité s'exprime dans d'autres règles qui, non impératives, relèvent, on le notera, d'une stylistique : significativité des noms donnés aux variables, aux constantes et aux procédures ; recours aux procédures et aux fonctions dont la déclaration préalable permet de mieux comprendre la partie exécutive.

Le rôle de la partie déclarative est par ailleurs double : du point de vue de la machine allocation d'emplacements en mémoire ; du point de vue du programmeur indications à caractère métalinguistique et métatextuel. C'est dans cette partie que l'utilisateur construit une extension du langage, c'est là que se manifeste le caractère non fini du LP, dont le lexique de base est complété par un vocabulaire nouveau. Si les mots dits réservés ne peuvent être touchés, l'utilisateur peut donner un sens nouveau au reste du lexique, il peut construire ses propres identificateurs - mots servant à désigner étiquettes, types, constantes, variables, fonctions et procédures. Ce sont évidemment les fonctions et les procédures qui manifestent de la manière la plus spectaculaire l'enrichissement du LP, dans la mesure où elles peuvent se substituer à une séquence longue et complexe d'instructions. Une procédure pouvant s'appeler elle-même, une récursivité tant langagière qu'algorithmique est présente par ce biais. La mise en place de modules qui peuvent être appelés par différents programmes contribue très largement aussi à l'enrichissement du vocabulaire. A noter que ces termes nouveaux élargissent aussi d'un certain point de vue la syntaxe du langage, puisque qu'une fonction, par exemple, comporte un certain type de paramètres, dans un certain ordre. Enfin l'innovation lexicale permet de submerger les termes anglo-saxons par des termes français évidemment plus transparents.

Une ponctuation énonciative permet de dissocier dans un programme des éléments qui n'ont pas le même destinataire : c'est ainsi que sont distingués du programme lui-même les commentaires en langue naturelle à destination humaine, encadrés de *{ }* ou de *(* *)* et les directives de compilation : encadrés des même signes, le premier étant suivi du signe *$*. D'autre part l'apostrophe signale le début et la fin des éléments de langue naturelle traités par le programme en tant qu'objets. Ce signe qui est une des variantes des guillemets anglo-saxons pose des problèmes pour le français, où l'apostrophe est fréquente : pour l'utiliser, il faut la redoubler, d'où, si l'on veut l'insérer dans une instruction *write*, la formulation *write('''');*

Sur un autre plan, l'existence de procédures et de fonctions complexifie l'énonciation : différents niveaux de discours apparaissent,

dans la mesure où il doit y avoir transmission d'informations entre les procédures de différents niveaux ; dans une procédure peuvent figurer trois sortes de variables : déclarées dans l'en-tête, précédées de *var*, elles sont assimilées aux variables du même nom du programme appelant et les valeurs qui leur sont données seront renvoyées vers ce programme ; déclarées dans l'en-tête mais non précédées de *var*, elles prennent comme valeurs celles des variables du programme appelant mentionnées dans l'instruction d'affectations, mais les changements internes à la procédure n'ont pas d'influence sur les valeurs de ces variables dans le programme principal ; enfin, les variables extérieures à l'en-tête sont purement locales. Ces différences aboutissent à créer une multiplicité d'énonciateurs et des règles de dialogue entre eux : c'est comme si ils ne parlaient pas complètement la même langue et ne disposaient pas des mêmes informations.

STRUCTURES

Revenons à des caractéristiques strictement structurales. Un LP est un système linguistique dépourvu de forme orale ; mais il est caractérisé comme les LN, par la double articulation : les unités significatives ou mots se composent de caractères, unités purement distinctives. Ces caractères sont ceux du code ASCII de 127 caractères, n'apparaissent donc ni les lettres accentuées ou cédillées ni les caractères graphiques. Ce n'est qu'en tant qu'éléments de langage-objet, dans des chaînes de caractères à traiter, qu'ils peuvent apparaître, c'est vrai aussi d'ailleurs pour les caractères de contrôle (0 à 31). Dans le LP, l'opposition minuscules-majuscules est neutralisée et les identificateurs peuvent comporter le signe souligné et des chiffres (rien n'empêche de donner à une variable le nom *A2_passionnément*. A titre d'opérateurs et de signes de ponctuation, sont utilisés divers symboles (+ - ^ *$* etc).

Rappelons aussi que le LP peut intégrer des nombres, avec certaines restrictions déterminées notamment par la mémoire (en LN aussi, un texte peut comporter des fragments mathématiques).

Les mots sont séparés par l'espace (chr(32)), les signes de ponctuation ou le retour à ligne (chr(13)). On peut se demander ce qui est l'équivalent du groupe de mots ou syntagme. L'expression semble en être un, en effet elle relie par des opérateurs arithmétiques ou logiques des opérandes qui sont des variantes ou des constantes, afin de donner une valeur au moment de l'exécution.

```
age := ancourant - andenaissance;
```

Éventuellement les parenthèses sont utilisés pour délimiter des expressions complexes, notamment booléennes :

```
if (age < 18) or (etat = incapable) or (etat = dechu) then
    electeur := false;
```

Les parenthèses sont d'ailleurs également utilisées pour la déclaration de types scalaire ou ensemble :

```
type
jour = (lun, mar, mer, jeu, ven, sam, dim)
```

Mais elles interviennent aussi dans un groupe où il y a subordination, pour introduire l'élément subordonné : dans la partie déclarative pour annoncer les paramètres d'une fonction ou d'une procédure et dans la partie exécutive pour les affecter. Une procédure standard comme *write( )* évoque la structure verbe + compléments. A l'intérieur des parenthèses, la virgule est utilisée pour séparer les différents éléments.

```
{partie déclarative}
function plusS(mot : longuechaine) : longuechaine;
begin
plusS := mot + 's';
end;
function infinitif (rad : longuechaine; term : courtechaine) :
    longuechaine;
begin
  infinitif := rad + term;
end;

{partie exécutive}
write('donnez le radical :');
readln(rd);
write('donnez le groupe : ');
readln(gr);
{ ... affectation de 'er' à tr}
vb := infinitif(rd, tr);
```

Autre structure avec subordination, l'indice d'un scalaire, toujours entre parenthèses : *jour(1).*

Constatons que la ponctuation du LP mixe valeurs mathématiques et linguistiques. Cependant, comme en mathématiques, elle est plus présente et plus rigide que dans la langue naturelle, puisqu'elle est indispensable à la création de groupes.

L'équivalent des phrases est constitué d'énoncés délimités par le point-virgule (ils ne s'identifient pas aux lignes, comme en basic ou en fortran), soit déclarations, soit instructions. En dehors des en-tête de programmes, de fonctions et de procédures qui ont une structure plus complexe, les déclarations sont composées essentiellement de deux éléments séparés par *:* ou = , le premier étant l'identificateur ou les identificateurs (séparés par la virgule) à déclarer, le second les éléments - souvent déjà connus - par rapport auxquels ils sont définis. Les déclarations de constantes typées ont une structure ternaire : *identificateur : type = valeur*. Les mots réservés comme *type*, *var* ou *const* introduisent les déclarations ou suite de déclarations et constituent des espèces d'étiquettes extérieures.

```
type
  longuechaine = string[80];
  tableautrois = array[1..3] of longuechaine;
var
  mot : longuechaine;
const
  articlesing : tableautrois = ('l''', 'le', 'la');
```

Les instructions simples peuvent être de quatre types : instruction vide (*begin end;*), instruction *goto* (ex. *goto fin;*), appel à procédure (ex. *clrscr* {efface écran}). Les instructions structurées (composées, itératives, conditionnelles, *with*) peuvent se comparer à des phrases complexes, mais aussi à des paragraphes ; en effet, grâce à la souplesse des *begin...end*, le Pascal ne fait pas la différence entre

```
if age = 18 then writeln 'vous êtes majeur'
else writeln 'vous êtes mineur';
```

et

```
if age = 18 then
  begin
    writeln 'vous êtes majeur';
    writeln 'êtes-vous inscrit sur les listes électorales?';
    readln(reponse);
    {etc.}
  end
```

```
else
  begin
  ...
  end;
```

La complexité, les niveaux d'emboîtements, comme pour les LN, ne sont pas formellement limités ; les limites existent moins au niveau de la compétence (du système) que de la performance (lisibilité, largeur de l'écran). D'autre part, s'il y a univocité des énoncés, ce n'est que dans une seule direction : si un énoncé n'a qu'un sens, un même sens peut s'exprimer dans plusieurs énoncés. La paraphrase est très présente dans les LP ; par exemple, la diversité des instructions itératives permet l'équivalence sémantico-pragmatique des deux formulations qui suivent :

```
while rep <> 'A' do
  begin
    write('donnez votre mot : ');
    readln(mot);
    write('tapez sur "A" pour arrêter, une autre touche pour
    continuer');
    read(kbd, rep);
end;
repeat
   write('donnez votre mot : ');
   readln(mot);
   {...}
until rep = 'A';
```

Un autre phénomène typique des LN se retrouve dans les LP : l'ellipse. Elle apparaît notamment dans l'instruction *with*, qui rend implicite le nom de l'enregistrement :

```
with auteur do
  begin
    write('donnez le nom de l''auteur : ');
    readln(nom);      {au lieu de auteur.nom}
    write('son prénom : ');
    readln(prenom);
    write('sa nationalité : ');
    readln(nationalite);
  end.
```

et dans l'instruction conditionnelle case :

```
case rep of 'E' :     {si la réponse est "E"}
  ecriture;
                  'L' :        {[si elle est] "L"}
  lecture;
                  'Q' :
  quitter;
end;
```

CONCLUSION

Résumons-nous : le Pascal, élaboré par des concepteurs humains pour faciliter la transmission des instructions données à la machine, reflète certaines caractéristiques essentielles des langues humaines, mais s'en distingue aussi par sa pauvreté, sa rigidité, sa spécialisation et sa sujétion étroite au principe de réalité, plus spécifiquement aux principes d'organisation des ordinateurs conçus par Von Neumann. A leur manière, tous les LP s'inscrivent dans ces limites et leurs différences sont toutes relatives.

V. Prince(6) écrit à ce sujet : *« l'apparente diversité des langages s'inscrit sur le même axe "données-traitements". En d'autres termes, si les langages procéduraux, calqués sur l'architecture organique de la machine, font davantage apparaître la séquentialité et la linéarité de leur démarche, les langages dits d'intelligence artificielle mettent plus en avant la vision atomiste de l'algèbre de Boole. Par conséquent, il nous semble excessif de penser qu'il y a un changement épistémique fondamental entre les deux... »* Tout au plus pourra-t-on préférer pour une initiation à l'informatique linguistique un langage procédural, qui met en lumière les limites des ordinateurs, notamment la démarche séquentielle évoquée plus haut et par conséquent une modularité qui, si elle s'inscrit dans une tradition rationaliste bien enracinée, ne peut rendre compte que très partiellement des stratégies langagières des êtres humains.

Jacques ANIS
Paris X Nanterre

(6) Dans un article passionnant intitulé « Épistémologie spontanée des traitements automatiques du langage », in *Histoire, Épistémologie, Langage*, tome 11, fascicule I, 1989, p. 105-126.