BASIC CCUPB: UN PAS VERS LA GENERACIÓ AUTOMÀTICA DE TRADUCTORS

M. BERTRAN F. BANQUE J.M. PORCAR

Amb la motivació de construir eines que facilitin la programació de traductors i de compiladors, el present article versa sobre l'estructura d'un traductor codificat en un llenguatge d'alt nivell (FORTRAN). L'estructura d'aquest traductor està preparada per a ésser generada automàticament partint d'un llenguatge d'alt nivell basat en el meta-llenguatge TBNF, versió modificada de la BNF.

Com a preparació es revisa la definició formal de llenguatges i les diverses tècniques de traductors, esmentant els mètodes de generació automàtica de traductors que es corresponen amb les tècniques de traductors existents.

L'exemple del BASIC construit al CCUPB ens serveix per a descriure l'estructura dels traductors. S'inclou també les descripcions de les màquines ideals BASIC executora i llistadora.

El BASIC-CCUPB forma part de la línia d'investigació de mètodes de construcció de traductors que es porta a terme al CCUPB.

L'article conté també referències rel.levants als conceptes presentats.

1. INTRODUCCIO

La proporció del cost del "software" és avui dia cada cop mes gran dins el cost total del disseny, depuració i posta a punt de la majoria de les aplicacions dels ordenadors.

La creació d'eines que simplifiquin el procés de generació de programes és doncs un objectiu important que condueix a l'abaratiment del "software". Un producte "software" és en molts casos un producte viu. Molt sovint s'hi afegiran noves opcions, extensions, i fins molts cops s'hi corregiran faltes no detectades durant el disseny.

Es doncs important des de aquest punt, que aquestes operacions siguin poc costoses de portar a terme. L'ús de llenguatges d'alt nivell adaptats a l'aplicació concreta és molt avantatjós ja que accelera el procés de posta a punt, modificació i correcció de les aplicacions.

Un llenguatge d'alt nivell ben dissenyat permet d'expressar d'una forma clara els algorismes que formen un programa. El llistat font és, en certa manera, part important del catàleg del "software". L'estudi i comprensió ràpides es faciliten notablement amb l'ús de llenguatges d'alt nivell. Com a contrapartida, l'ús d'aquests llenguatges no permet una utilització tant "eficient" del temps de processador ni de la memòria. Amb tot, la eficiència s'ha de mesurar globalment tenint en compte no sols aquests dos factors de cost sinó el cost de posta a punt, el de la introducció de modificacions i el de les correccions durant la vida del producte complet.

El present article versa sobre el problema de la generació automàtica de traductors de llenguatges d'alt nivell. Després d'una breu exposició de les dues tècniques principals, s'aborda la coneguda amb el nom de "top-down" /1/. La descripció es fa sobre un exemple: El traductor del BASIC CCUPB /2/. La implementació actual d'aquest producte és un pas intermedi vers la construcció d'un llenguatge d'alt nivell dedicat a la generació de traductors. El traductor està també construït en un llenguatge d'alt nivell (FORTRAN), però la seva estructura és pensada per a constituir el resultat de la traducció d'un font basat en el meta-llenguatge BNF /3/.

La descripció, tant de la màquina BASIC interpretadora com del traductor de BASIC, per

- M. Bertrán, F. Banqué i J.M. Porcar del Centre de Càlcul de la Universitat Politècnica de Barcelona.
- Article rebut el Agost.

met d'exposat una tècnica d'estructuració de traductors.

2. DESCRIPCIÓ FORMAL DE LLENGUATGES

Un conjunt ordenat de símbols formen una sentència. Un conjunt determinat de sentències vàlides formen un llenguatge. Els símbols que formen una sentència els nomenarem terminals. Es útil d'emprar símbols que mai no apareixeran dins d'una sentència vàlida però que representen certs conjunts de terminals. Els símbols no-terminals representaran doncs grups de terminals i en general de terminals i no-terminals. En particular, el símbol no terminal <s> representarà totes les sentències vàlides del llenguatge.

La sentència de BASIC font:

```
100  IF  A + B > C   THEN   10
```

és formada pels símbols terminals següents: 1, 0, IF, A, +, B, >, C, THEN. Observem doncs que tant les seqüències de caràcters IF com THEN són considerades com un sol símbol terminal. Per conveniència es pot introduir el símbol no-terminal representatiu de tota expressió aritmètica en el nostre llenguatge: <expr.arit>. D'una forma semblant, el símbol no terminal <num.sent> definirà tots els números de sentència vàlids. Si es fa ús d'aquests dos símbols es pot obtenir una representació més general de la nostra sentència:

```
<num.sent> IF<expr.arit> > <expr.arit> THEN
<num.sent>
```

Es comprèn que introduint nous símbols no terminals es pot anar generalitzant la descripció. Per exemple, la introducció dels no-terminals <fórmula> per a representar una operació lògica i <cos if> per a representar el cos de la sentència IF ens porta a les dues representacions següents:

```
<num.sent> IF<fórmula> THEN <num.sent>
<num.sent> IF<cos if>
```

La introducció de nous símbols permet d'arribar al símbol objectiu: <s> que representa tota sentència vàlida.

S'ha convertit doncs una sentència vàlida en el símbol <s> mitjançant una seqüència de passos. A cada pas s'ha fet ús de substitucions. Per exemple: "A+B" per <expr.arit> o bé "100" per <num.sent>. Per a especificar el nostre llenguatge ens falta doncs definir les regles de substitució o produccions. Es fa ús de la notació BNF /3/.

Les produccions seran bàsicament una taula de dues entrades. L'esquerra contindrà un símbol no-terminal i la dreta la seqüencia de terminals i no-terminals que pot substituir el símbol de l'esquerra. Per exemple:

```
 1 <s>          ::= <num.sent><sent.>
 2 <sent.>      ::= IF <cos if>
 3                | SUB <cos sub.>
 4                  LET <cos let>
 5 <cos if>     ::= <fórmula> THEN <num.sent>
 6 <fórmula>    ::= <expr.arit><rel.log>
                    <expr.arit>
 7 <rel.log>    ::= >
 8                | <
 9                | =
10                | ≠
11                | ¬ =
12 <num.sent>   ::= (número enter)
13 <expr.arit>  ::= (expressió aritmètica vàlida)
```

En el primer pas de la conversió de la sentència IF s'ha fet ús de les produccions 12 i 13. En el segon s'ha introduït la 6 i la 7. Finalment, les produccions 5, 2 i 1 porten al símbol <s>.

La substitució dels símbols de l'esquerra es pot fer sempre que el símbol aparegui en una forma intermèdia, independentment dels símbols veïns o del context. Si bé hi ha llenguatges on això no és cert (sensibles al context), la sintaxi de la gran mayoria de llenguatges de programació és insensible al context.

3. TRADUCTORS I GENERADORS DE TRADUCTORS

Un traductor ha de reconèixer en primer lloc si el conjunt de símbols a traduir formen una sentència vàlida. La part sintàctica del traductor es reduirà doncs a determinar una seqüencia de produccions que permetin de passar del conjunt de símbols d'entrada al símbol <s>.

Si aquesta seqüència existeix, la sentència serà considerada correcta. Hi han dues formes de portar a terme aquest procés: de dalt a baix (Top-Down) o bé de baix a dalt (Bottom-Up).

L'exemple descrit al principi correspon a la segona tècnica. El traductor comença l'anàlisi per el símbol de l'esquerra i avança vers la dreta. En aquest procés, i guiat per la taula de produccions, determina si pot efectuar una substitució. Si el nou símbol permet d'aplicar una producció, el traductor farà la substitució corresponent reduint el conjunt de símbols en introduir el nou no-terminal. En un instant donat de la traducció tindrem doncs la sentència dividida en tres parts: porció reduïda, integrada per terminals i no-terminals; símbols analitzats i encara no sustituïts; finalment text encara no analitzat.

```
                          ↓
<num.sent>IF<expr.arit> > C  THEN 10
 Porció reduïda (stack)  |   Text no
                         |   analitzat.
                         └── Cap de text.
```

Per a decidir la producció a aplicar, el traductor activat de baix a dalt necessita doncs una taula de dues entrades que segons els símbols últims de la porció reduïda i els símbols del cap del text li permeti d'identificar la producció que ha d'aplicar.

L'aspecte semàntic del traductor, o sia la generació de símbols corresponents al llenguatge objecte, es porta a terme en cada instant en què el traductor decideix d'aplicar una producció. Per exemple, en el moment que la producció 5 és aplicada, hom està en condicions d'afegir a l'objecte una instrucció de salt condicional. El traductor sap el símbol lògic perquè ja ha estat analitzat prèviament i ja ha generat instruccions per a avaluar les dues expressions aritmètiques i la fórmula lògica en aplicar abans la producció 6.

Existeixen eines per a la generació automàtica de traductors activats de baix a dalt. Aquestes tècniques, partint de la descripció BNF de la sintaxi del llenguatge, generen la taula de dues entrades de l'analitzador sintàctic que permetrà de determinar les produccions a aplicar en cada pas de la traducció. La generació de codi objecte la fa una subrutina que l'usuari ha de construir d'acord amb la seva descripció sintàctica i amb la màquina objecte (veure /4/).

El primer tipus de traductor procedeix a la inversa: va de dalt a baix (Top-Down). Partint del símbol objecte <s> i analitzant el text font, també d'esquerra a dreta, prova en cada moment d'aplicar les produccions que permetin de substituir <s> pel text font. Si la substitució és possible aplicant un nombre finit de produccions, la sentència és acceptada. Per tal de poder decidir, haurà de veure abans si pot aplicar alguna producció que permeti de substituir els no-terminals constitutius de la primera producció. Això el portarà a considerar noves produccions fins a arribar a una que conté terminals que són iguals als primers símbols del text. Això permetrà d'avançar, al traductor, en el procés d'anàlisi. A l'instant, doncs, que un determinat no-terminal existent a la part dreta d'alguna producció ha estat trobat com a vàlid, el traductor pot passar a determinar si el següent símbol dins la mateixa producció és també vàlid, en definitiva, la producció de substitució d'aquest símbol no-terminal juntament amb els següents símbols del text que encara no ha estat analitzat.

Hi ha moments que tenint en compte el següent símbol del text i el no-terminal que el traductor està analitzant, dirigit per l'estructura de les produccions de substitució del no-terminal, el traductor pot decidir que el no-terminal no es pot formar amb els següents símbols. Llavors el traductor decideix que aquest no-terminal és invàlid i passa a considerar noves possibilitats. Aquestes possibilitats vénen donades per produccions diferentes que poden substituir el mateix no-terminal (per exemple: les produccions 2, 3 i 4 o bé les 7, 8, 9, 10 i 11 en el primer exemple).

Observem que el conjunt de produccions BNF del llenguatge que volem traduir constitueix, no tan sols la especificació sintàctica, sinó també el mateix programa analitzador sintàctic. Aquest programa és, però, codificat en un llenguatge de molt alt nivell. Efectivament, tenim tan sols d'associar a tota aparició d'un no-terminal a la dreta del símbol "::=" una crida a una subrutina. Cada produc

ció és doncs una subrutina que s'executa començant per el símbol "::=" d'esquerra a dreta. La aparició d'un terminal s'ha d'associar també a una subrutina que consisteix a determinar si existeix o no el símbol terminal en qüestió, sempre dins els següents símbols no analitzats del text.

Hi ha també d'altres convencions: a l'entrada a qualsevol producció o subrutina, es guarda l'índex que indica la posició dins del text del símbol següent que toca analitzar. Durant l'execució de la subrutina aquest índex anirà creixent. Si la sortida de la subrutina correspon a no haver trobat la seqüència de símbols terminals i no-terminals, constitutius de la subrutina, es retorna la condició invàlid i es restitueix l'índex de text al valor que tenia en el moment d'entrada a la subrutina corresponent. Sempre, naturalment, que la invalidesa no indiqui un error en el text, com per exemple, a la producció 5, si després d'haver trobat THEN, el retorn de <num.sent> es invàlid.

El retorn serà vàlid si s'han arribat a detectar com a vàlids tots els símbols que apareixen a la producció. Finalment, el retorn vàlid de la primera producció de l'exemple anterior correspon al reconeixement d'una sentència font vàlida.

La part semàntica del traductor activat de dalt a baix consisteix també en la generació del codi objecte en el moment de reconeixement de les produccions. Les accions semàntiques poden ésser de molts tipus, des de guardar un operador o bé una variable per a fer-ne ús posteriorment fins a generar codi objecte. Una forma còmoda de treballar és ampliar el llenguatge de molt alt nivell, que fins ara era vàlid per a l'analitzador, insertant accions entre els símbols terminals i no terminals de les produccions. Aquestes accions s'interpretaran també com a crides a subrutines que portaran a terme les operaciones de generar codi objecte, generar símbols corresponents a determinades variables, etc...

Concretarem aquest punt referint-nos novament a l'exemple de sempre. Si introduïn les accions A1(n), A2(op,k) i A3(k) per a generar un salt condicional a n (codi SS) segons l'estat lògic de l'acumulador (veritat o fals); per a generar el codi objecte representatiu d'una operació lògica (L>, L<, etc...) entre l'acumulador i una posició de memòria (k) deixant el resultat a l'acumulador; i per a generar una instrucció d'emmagatzemament (ST) a la posició de memòria k respectivament, la part sintàctica corresponent a les produccions 5 i 6 es transforma, en introduir la part semàntica, en

```
5 <cos if>  ::= <fórmula> THEN<num.sent>A1(n)
6 <fórmula> ::= <expr.arit> A3(k)<rel.log>
      <expr.arit> A2(op,k)
```

El traductor entrarà a la subrutina <fórmula> activat per l'aparició del no-terminal <fórmula> a la subrutina <cos if>, després de reconèixer <expr.arit>, generant el codi que deixarà el resultat a l'acumulador durant l'execució, l'acció A3(k) generarà una instrucció ST,k per a emmagatzemar el contingut de l'acumulador a la posició k de la memòria. El reconeixement de <rel.log> com a vàlid comporta de guardar l'operador a la variable "op" del traductor. Finalment, el retorn vàlid de l'últim <expr.arit>, implicant la generació de codi objecte que deixarà el resultat de la segona expressió aritmètica a l'acumulador durant l'execució, activa l'acció A2(op,k) que genera la instrucció L>,k per a comparar el contingut de la posició de memòria k amb el valor de l'acumulador deixant l'acumulador en estat de veritat o de falsedat. El retorn vàlid de <fórmula> activa THEN que troba aquest terminal dins el text i finalment <num.sent> que deixa el número de sentència a la variable n, activant l'acció A1(n) que genera el salt condicional a l'adreça n.

```
[ ]   Codi per a avaluar la 1a. expressió
[ ]   aritmètica a l'acumulador.
ST k  Guardar l'acumulador a la posició k
[ ]   Codi per a avaluar la 2a. expressió
[ ]   aritmètica a l'acumulador.
L k   Comparació amb la posició k.
SS n  Salt condicional a n segons l'acumulador.
```

A la figura podem observar l'estructura del codi generat pel traductor de la sentència IF.

Fem notar finalment que la generació automàtica de traductors activats de dalt a baix consisteix bàsicament a codificar el traductor en el llenguatge de molt alt nivell que acabem d'introduir. Un programa especial -- traduirà aquest font en codi executable.

4. ESTRUCTURA DE LA MÀQUINA BASIC

Abans de passar a descriure el traductor BASIC, i com a preparació expliquem l'estructira general i l'organització de la màquina BASIC. El llenguatge Basic Intermedi (BI), generat pel traductor, és l'eix del sistema BASIC, i sobre ell treballen les màquines - BASIC executora (MBE) i Basic llistadora -- (MBL), que passem a descriure tot seguit. - La característica comuna d'ambues màquines es que són màquines "stack" o "pila", entenent per "pila" una cua que creix i decreix sempre pel mateix costat, i que per tant el primer que entra es el primer que surt. Les màquines basades en piles ofereixen una bona solució al problema de assignació de memòria per variables auxiliars que contenen resultats parcials (aritmètics o bé de control). Així és possible de resoldre qualsevol tipus de recursivitat que es doni dins el programa. Per exemple, una rutina es pot cridar a ella mateixa tantes vegades com -- calgui, quedant tant els resultats parcials com les adreces de retorn en el lloc apro-- piat de la pila.

4.1 Màquina Basic Executora (MBE)

Aquesta màquina parteix del llenguatge intermedi i té per missió l'execució sobre la pila de les diferents instruccions del BI. -- Cal remarcar que la MBE es bàsicament una - màquina d'interpretació seqüencial que treballa sobre la pila d'execució, sobre taules com són la de variables, o s'emmagatzema el valor de cada variable (VAR), la de - matrius on hi ha informació referida a les matrius del programa (MAT) i sobre un conjunt de piles auxiliars com la d'equivalències, que permet de conèixer l'equivalència d'una variable entre la rutina i el principal, la de bucles, etc...

Un parell d'exemples ens poden ajudar a comprendre el mecanisme de la MBE.

Considerem la sentència Basic Font:

```
100   IF A+B>C THEN 10
```

El traductor de Basic, genera el següent codi:

```
FV,A,FV,B,AD,FV,C,GT,SS,10
```

I la execució és:

FV,A: Trobar el valor de la variable A -- (dins VAR) i posar-lo a la pila.

FV,B: El mateix procés per a la variable - B.

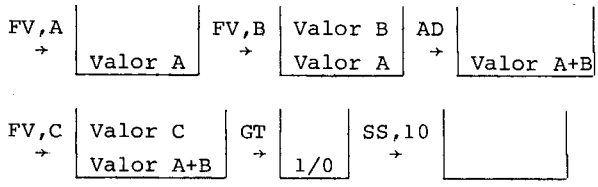
AD: Sumar les dues últimes entrades de - la pila posant el resultat en lloc - de l'última entrada de la pila.

FV,C: El mateix procés que per a la variable A.

GT: Comparació. Si l'última entrada de - la pila i la penúltima compleixen la condició es posa un 1 a l'última entrada de la pila. En cas contrari -- s'hi posa un zero.

SS,10: Salt condicional. Salta a 10 si al - capdamunt de la pila hi ha un 1, i - no salta si ha un zero.

L'evolució de la pila és doncs:



El segon exemple explica l'ús de la pila -- d'equivalències i de referències, anomena-- des abans. Suposem la seqüència de sentèn-- cies BF següent:

```
200  GOSUB 300,A,B,A B
       :
300  SUB A,X,Y
       :
320  RETURN
```

En executar-se la primera sentència es fa - el següent:

- Es passen els valors de A, B i A+B, a la dreta de la pila d'equivalències.
- Es fa una nova entrada a la pila de referència amb el nombre de paràmetres que -- s'han passat.
- S'incrementa el nivell de subrutina.

L'execució de la segona sentència representa:

- Passar a l'esquerra de la pila d'equiva-- lències les adreces de les variables que són paràmetres de pas. Amb això tota referència a una variable queda convertida automàticament en referència a la seva va-- riable equivalent.

L'execució de la sentència RETURN fa que es passi els valors de la dreta de la pila d'equivalències al seu lloc a VAR, es desfan les entrades de la pila d'equivalències i de referència i es decrementa el nivell de subrutina. L'estat de les piles d'equivalència i de referència quan l'execució de la rutina és el següent:

NVLSUB
2

STKRF
3
0

	STKEQ
Y	Valor A+B
X	Valor B
A	Valor A

La pila d'equivalències, ens indica, doncs, l'equivalència de cadascún dels paràmetres de la subrutina dins i fora d'aquesta. La pila de referència indica per cada crida a una subrutina l'últim valor de la pila d'equivalències que correspon a la crida. Per diferència entre les dues darreres entrades d'aquesta pila es pot trobar el nombre de paràmetres. La variable NVLSUB indica el nivell de subrutina.

Una explicació detallada de totes les piles, el seu funcionament i del conjunt sencer de instruccions de BI i del seu significat, el podem trobar a /2/.

4.2 Màquina Basic Llistadora (MBL)

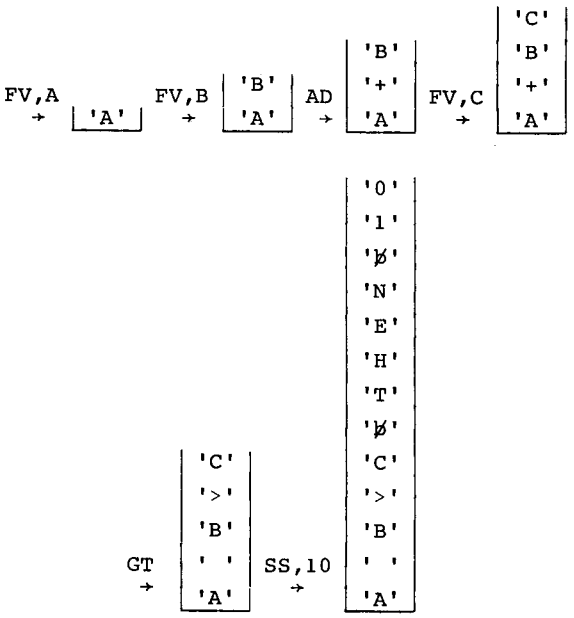
La màquina Basic llistadora forma amb la ja explicada MBE, el cos de la màquina BASIC. Parteix del mateix llenguatge intermedi BI que ha generat el traductor. La pila es fa servir en aquest cas per a posar-hi els caràcters a llistar a mesura que es van generant, de forma que en acabar la seqüència d'instruccions que formen la sentència, la pila conté tota la sentència a punt d'ésser llistada.

El mateix exemple utilitzat abans servirà per a explicar la MBL

```
100  IF A+B>C THEN 10
```

que genera:

FV,A,FV,B,AD,FV,C,GT,SS,10



El tipus de sentència així com el número es coneixen de forma independent, car són uns punts fixos i no sotmesos a l'estratègia de piles.

Per a cada instrucció de BI hi ha un tractament paral.lel al de la MBE que permet de generar codi de llistat de la part de la -- sentència a què correspon la instrucció.

El punt de la màquina llistadora més interessant a considerar és el referent a la inserció o no de parèntesis. Per a saber si cal englobar o no dins parèntesis dues entrades de la pila de llistat (que són dos conjunts de caràcters que formen blocs ja resolts) es forma i es va consultat una pila d'operadors, que conté per a cada entrada de la pila de llistat una indicació de l'operador de més alt nivell segons una definició de nivells de prioritat semblant a la d'altres llenguatges /3/. Si cal, en resoldre un bloc, s'inserten parèntesis i de

totes passades s'actualitza la pila d'operadors.

5. TRADUCTOR DE BASIC

El traductor de la implementació actual de Basic és de tipus d'activació de dalt a baix i és codificat en llenguatge FORTRAN. Com s'ha vist abans, aquest tipus de traductors poden ésser programats en un llenguatge que és una extensió de la BNF. En el present cas s'ha fet ús de la TBNF /5/. Si traduïm aquest programa a un llenguatge entès per una màquina física tindrem el programa traductor executable. Aquesta traducció es podria fer automàticament. Per a construir un programa traductor caldria tan sols programar-lo en TBNF i activar un traductor especial que generés el programa traductor executable a partir del mateix algorisme expressat en TBNF.

En el present cas, després d'expressar el traductor de Basic en TBNF, és traduït a FORTRAN manualment. La versió FORTRAN està però estructurada perquè en un futur es pogués fer la traducció TBNF a FORTRAN d'una forma automàtica.

Les subrutines corresponents als símbols no-terminals que constitueixen un traductor activat de dalt a baix, han d'ésser en general recursives. Això és degut al fet que molts cops un símbol no-terminal apareix a la part dreta de la seva producció o bé d'alguna producció activada per no-terminals constitutius de les produccions del primer no-terminal. Un exemple d'aquesta situació es té en la definició BNF d'una llista d'expressions aritmètiques:

```
<llist.expr> ::= <expr.arit>, <llist.expr>
               | <expr.arit>
```

on el no-terminal <llist.expr> apareix a la dreta de la seva producció. En conseqüència, una codificació FORTRAN d'un traductor ha de permetre que moltes de les seves rutines siguin recursives.

El problema de la recursivitat i la seva implementació és tractat per diversos autors /6/. S'ha adoptat la tècnica de la sentència GO TO calculat (GO TO (1,2..,4,..),I).

Tota rutina recursiva és part del programa principal. El traductor fa ús d'una pila. Abans de transferir el control a la subrutina recursiva 1, amb una sentència GO TO, col.loca al cap de la pila l'etiqueta r (PUSHS(r)) de la següent sentència. En el retorn d'una subrutina recursiva, es recupera l'etiqueta r del cap de la pila (POP(I)) i es fa servir d'índex dins d'un GO TO calculat:

```
                        ........
(1) Crida:              CALL PUSHS (r)
                        GO TO 1
                      r CONTINUE
                        ........
(2) Entrada de
    subrutina:        1 CONTINUE
                        ........
(3) Retorn de la        POP (I)
    subrutina:          GO TO (1,2,...,4...),I
```

Cada subrutina tindrà doncs una etiqueta de sentència FORTRAN. Tota aparició d'un terminal a la part dreta d'una producció correspondrà en el programa FORTRAN al grup de crida (1). Al final de tota producció es farà transferència al grup de retorn (3) situat al final del programa principal.

Amb l'objecte d'expossar l'estil de la construcció del tràductor, basat com s'ha dit en la TBNF, es farà ús de les sentències de salt i d'entrada a subrutina:

```
  10  GOSUB 1000 ; A,B,A+B
      ......
1000  SUB X,Y,Z
```

on el pas de paràmetres és opcional. L'especificació TBNF és la següent:

```
20  <sent> ::= ("GOSUB"<cos.gosub>|"SUB"
               <cos.sub>|..)
21  <cos.gosub> ::= <num.sent>A21 OPT(";"
               <llist.expr>A25) A23
22  <cos.sub> ::= <llist.simb> A27
23  <llist.expr> ::= DSEQ 1-...(<expr.arit>
               A24) ","
24  <llist.simb> ::= DSEQ 0-...(<símbol>A26)
               ","
```

Les accions tenen el significat següent:

A21: Guardar <num.sent>.
A23: Generar PS,<num.sent>.

A24: Incrementar el conta de expressions: - m ← m+1.
A25: Generar TD,m.
A26: Generar FD,<símbol>. Incrementar el -- conta de símbols: n ← n 1.
A27: Generar TI,n.

En primer lloc s'explica el significat dels operadors i construccions TBNF que s'usen - en la especificació exemple.

La construcció: (xxx...xx | yyy...yy | ...) equival a la operació lògica "o". Tant el - grup xxx...xx com el grup yyy...yy com els altres són correctes en aquest punt. Però - tan sols un d'ells. El traductor comença -- per buscar el grup xxx...xx. Si el troba -- surt de la construcció. Si no el troba passa a bucar el següent grup (yyy...yy) i fa el mateix.

Els terminals es limiten amb comes per a -- distingir-los dels operadors de TBNF. Els - que apareixen en aquesta descripció son -- DSEQ k-l i OPT.

L'operador DSEQ indica la validesa d'una seqüència d'un mínim de k i un màxim de l, de grups com el que segueix primer, separats - pel grup que segueix segon. Observem com -- l'ús de l'operador DSEQ per a definir una - llista d'expressions (<llist.expr>) ens estalvia una producció i elimina la recursivitat de <llist.expr>.

L'operador OPT indica aparició opcional del grup següent. Equival a DSEQ 0-l.

El Basic objecte generat per les produccions 20, 21,...,24 del traductor activat per les sentències: 10 i 1000 és el següent:

```
10: ∫ \ Codi per a avaluar A deixant el --
    \ ∫ seu valor en el cap de la pila.
    ----
    ∫ \ Codi per a avaluar B deixant el --
    \ ∫ seu valor en el cap de la pila.
    ----
    ∫ \ Codi per a avaluar A+B deixant el
    \ ∫ seu valor en el cap de la pila.
    ----
     TD  Codi per a transferir els tres va-
      3  lors dels cap de la pila al cap de
         la pila d'equivalències.
    ----
```

PS Codi per a transferir control a 1000 - 1000 deixant en el cap de la pila l'adreça següent d'execució.

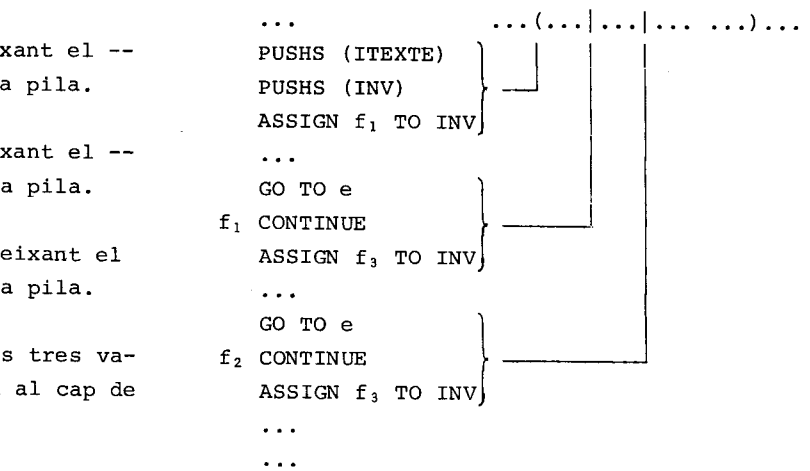
```
1000: FD  Codi per a deixar en el cap de la
       X  pila el símbol X.
      ----
       FD  Id. amb el símbol Y.
       Y
      ----
       FD  Id. amb el símbol Z.
       Z
      ----
       TI  Codi per a transferir els tres --
        3  símbols dels cap de la pila a les
           entrades de l'esquerra de la pila
           d'equivalències.
```

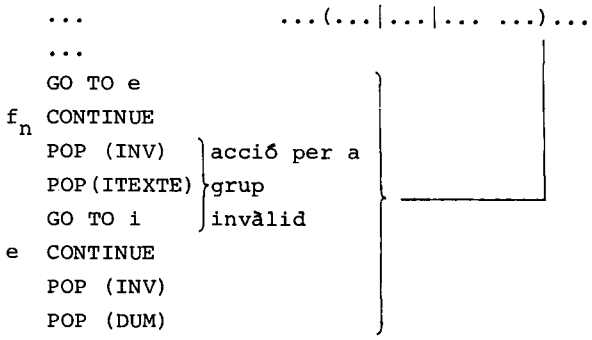
Tal com ho hem fet a la secció 2 amb el traductor de la sentència IF podríem també -- aquí seguir els passos del traductor de Basic programat en TBNF. El procés és molt -- semblant al descrit abans, tenint en compte, però, el significat de les noves construccions i operadors TBNF. Per tant, no es repeteix.

Per acabar la descripció de la implementació actual del traductor de Basic s'esmenten les construccions FORTRAN que equivalen als grups TBNF que apareixen a les produccions 20,21,...,24.

Al final del programa principal hi ha la -- sentència: i GO TO INV,(f_1,f_2, ...). Si es troba algun grup invàlid es bifurca sempre a aquesta sentència que dirigirà l'execució al punt convenient.

Construcció "o" lògic:





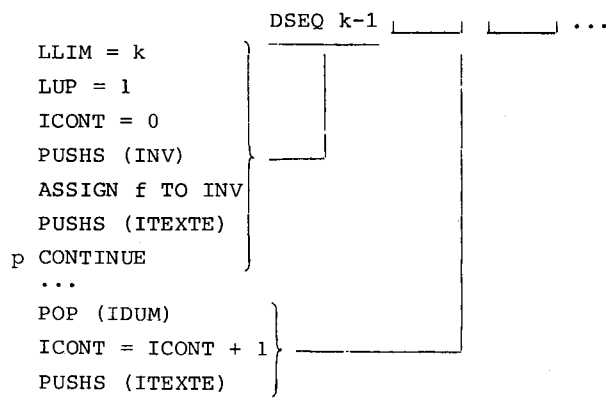
Construcció ...<nom>... :

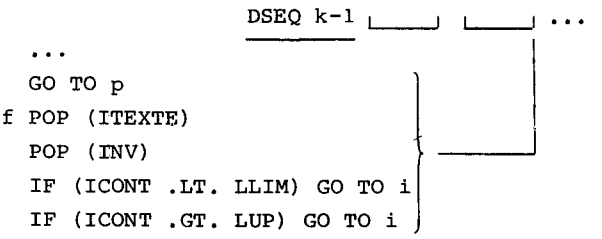
Ja hem descrit l'aspecte de transferència de control. S'afegeixen ara les operacions de - guardar i recuperar variables. Una d'elles - serà l'etiqueta de bifurcació en cas de trobar un grup invàlid (INV).

```
  ...
  PUSHS (ITEXTE)
  MPUSH (V1,V2,V3,V4)
  PUSHS (INV)
  PUSHS (r)
  GO TO 1
r CONTINUE
  POP (ICAP)
  POP (INV)
  MPOP (V1,V2,V3,V4)
  POP (IDUM)
  IF (ICAP .NE. FALS) GO TO c
       ITEXTE = IDUM
       GO TO i
c CONTINUE
  ...
```

Observem com en el moment de retorn de tota subrutina, el cap de la pila conté el valor FALS (=0) o bé un valor diferent de zero -- (operador, símbol, etc...).

Operador seqüència amb delimitadors.





S'observa com el traductor fa servir la pila per a guardar i recuperar en el moment - oportú l'índex de text i l'adreça de bifurcació en cas de grup invàlid.

6. CONCLUSIÓ I DIRECCIONS FUTURES

Després de descriure l'estructura d'un traductor activat de dalt a baix expressat en llenguatge FORTRAN, es pot observar clara-- ment la correspondència entre els grups de sentències FORTRAN i els símbols del llen-- guatge TBNF. Es pot doncs pensar en una màquina ideal: màquina traductora, les seves instruccions executen el mateix que cada -- grup de sentències FORTRAN. La màquina traductora és una màquina basada en una pila - que creix i decreix a mida que avança el -- procés de traducció.

El disseny d'un traductor que partint de -- TBNF generi codi per a aquesta màquina traductora facilitaria moltíssim la construcció i posta a punt de traductors en general. La màquina traductora és un interpretador cons truït sobre la màquina física disponible. - El disseny i construcció d'aquest generador de traductors podria ésser una línia d'in-- vestigació aplicada a empendre, que conduiria, entre d'altres resultats, a la reducció considerable de la memòria ocupada pel programa traductor.

7. REFERÈNCIES

/1/ GRIES, D. "Construcción de compiladores" Paraninfo. 1975.

/2/ BERTRAN, M., PORCAR, J.M., BANQUE, F. - "BASIC CCUPB Descripción preliminar". - Centre de Càlcul UPB. Maig 1976.

/3/ NAUR, P. et al. "Report on the algorith mic language ALGOL 60". Comm. ACM, 3 --

(1960). p. 299.

/4/ McKEEMAN, HORNING, W.H., WORTMAN, D.B. "A compiler generator". Prentice-Hall, 1970.

/5/ LAWSON, H.W., DOUCETTE, D.R. "A translation machine with automated Top-Down -- parsing". ACM Sigplan Notices, Feb.1976, V. 11, N. 2.

/6/ BROWN, D.W. "Recursive techniques in -- programming". American Elsevier, New -- York, 1975.