Universidad Nacional Autónoma de México

FACULTAD DE INGENIERIA

UN GENERADOR DE TABLAS DE CONTROL PARA UN PARSER TOP-DOWN PARA GRAMATICAS LL(1)

T E S I S

para obtener el Titulo de

INGENIERO EN COMPUTACION

P r e s e n t a

ALFREDO OCADIZ MENDOZA

México, D. F. *1985*

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

I N T R O D U C C I O N

La finalidad de este trabajo es presentar la utilización práctica de una serie de algoritmos, tales como: Eliminar Simbolos Inútiles, Producciones Sencillas, Producciones Epsilon, Recursión por la izquierda, Factorización y Generar la Tabla de Control; los cuales son aplicados a gramáticas libres de contexto en la teoría de compiladores. Para esto se pensó en su implementación dentro de un sistema manipulador de gramáticas, el cual acepta como entrada cualquier gramática que cumpla con las características de las gramáticas libres de contexto, produciendo como salida otra gramática equivalente.

Además, el sistema también puede determinar si la gramática equivalente es del tipo LL(1), siendo éste el caso, el sistema procede a generar su respectiva tabla de control, la cual es utilizada por el parser top-down, el cual a su vez se auxilia de un stack de estados, sobre el cual efectua ciertos movimientos de acuerdo a las rutinas contenidas en la tabla de control que ha sido generada. De aquí el parser top-down podrá determinar de su análisis efectuado sobre cualquier cadena que haya sido derivada a partir de la entrada si esta es aceptada o rechazada por el automata top-down.

Esta parte del sistema viene a formar lo que sería el analizador sintáctico o parser en la segunda fase del proceso de un compilador.

Es conveniente hacer notar la diferencia que existe entre el sistema manipulador de gramáticas y el compilador de compiladores, que se define como un programa que acepta como entrada una gramática, que es analizada y procesada por ciertas rutinas y reglas para desambiguarla en caso de serlo, o indicar precedencias, produciendo como salida un compilador.

En cambio el sistema manipulador de gramáticas, durante su proceso debe seguir cierta secuencia de pasos sobre la gramática a ser analizada, o sea que su procedimiento no es tan automático como el caso del compilador de compiladores.

Una diferencia importante entre el compilador y el compilador de compiladores es la presentan sus entradas y salidas.

Ya que el compilador es un programa traductor que recibe como entrada un programa fuente (en lenguaje de alto nivel) y produce como salida un programa objeto (en lenguaje de maquina).

A continuación se da una descripción en forma generalizada de la función que tiene un compilador de compiladores. Su primera fase se inicia con la ejecución del analizador léxicográfico o scanner, el cual analiza la entrada fuente o gramática, y reconoce los elementos basicos llamados tokens o símbolos terminales. Generalmente los tokens son: palabras reservadas, identificadores, operadores y simbolos de puntuación.

La salida producida por el scanner es tomada como entrada por la siguiente fase llamada analizador sintáctico o parser,

quien reconoce las estructuras sintacticas o simbolos no-terminales. El parser para su analisis utiliza un autómata de estados finitos sobre el cual realiza ciertas acciones; los estados de los simbolos terminales son representados por medio de etiquetas, asi una palabra reservada puede ser representada por el número 1, un operador por 2, un identificador por 3, etc.

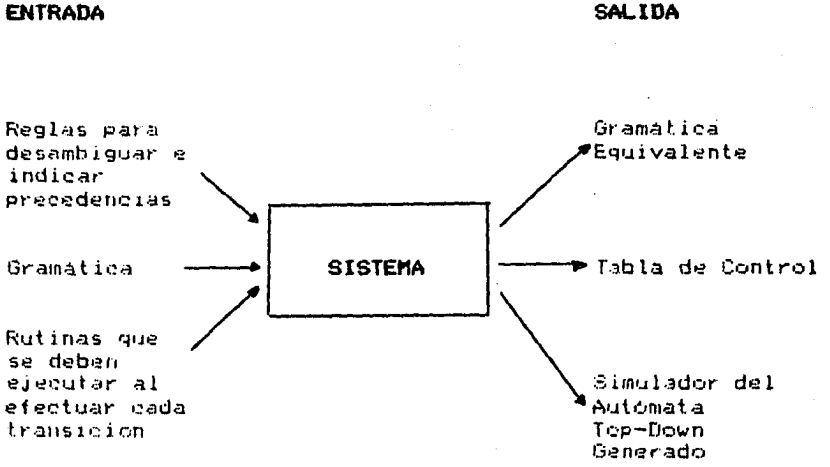
Como resultado de ésta fase, se determina si la cadena es aceptada o rechazada.

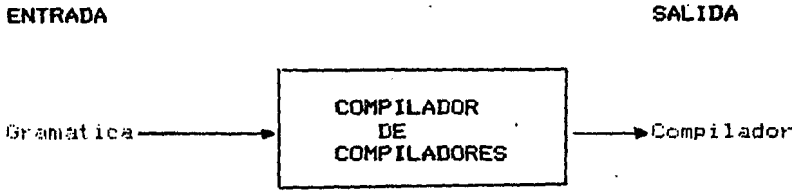
Dentro de la fase del parser se aplican ciertas reglas para desambiguar gramaticas e indicar precedencias. Las reglas de precedencia tienen como objetivo proporcionar los niveles de precedencia de los operadores para el caso de una expresión arimetica, asi como el orden que se debe seguir al ejecutar una instruccion.

Es importante mencionar ciertas rutinas que interactuan con todas las fases del compilador de compiladores, tal como la rutina de manejo de errores; la cual detecta y corrige errores de sintaxis.

Otra rutina es la que lleva a cabo el manejo de tablas, que hace uso de una estructura de datos llamada tabla de símbolos, la cual almacena todos los nombres utilizados por la entrada fuente.

El esquema que se presenta a continuacion, muestra la diferencia que existente entre el sistema manipulador de gramáticas y el compilador de compiladores.





En el capitulo siguiente se dan las características de las gramáticas libres de contexto, así como la definición de los conseptos que se aplican en capítulos posteriores; dentro de éste capítulo es menciona tambien el procedimiento seguido por cada uno de los algoritmos mencionados anteriormente, proporcionando a su vez algunos ejemplos para dar una mejor visión de lo que pretende cada algoritmo.

Generación de la tabla de control, capitulo que muestra en forma detallada los trece pasos que se siguen para su obtención, siendo el paso trece el que simula el parser top-down.

El último capítulo que comprende resultados y conclusiones, habla sobre las características principales del equipo de computación, en el cual fue desarrollado el sistema de información, así como la descripción del sistema y su uso práctico.

ALGUNAS CARACTERISTICAS DE LAS GRAMATICAS LIBRES DE CONTEXTO

1.- GRAMATICAS LIBRES DE CONTEXTO

Una gramática libre de contexto, se define como aquella gramática capaz de definir las características sintácticas de un lenguaje de programación, cuya estructura está formada por un número finito de símbolos no-terminales tal como: <DECLARACIONES>, <EXPRESIONES ARITMETICAS>, etc, especificadas en un lenguaje de programación.

Las palabras reservadas como: BEGIN, END, operadores; +, identificadores y símbolos de puntuación definen los símbolos terminales o tokens en una gramática.

Las reglas de la gramática algunas veces son llamadas producciones y son de la forma:

Cualquier no-terminal --> Cualquier secuencia de terminales y no-terminales

La secuencia del lado derecho de la flecha puede ser una secuencia nula. Un ejemplo de tal producción es:

$$\langle A \rangle \longrightarrow \epsilon$$

Este tipo de producción es referenciada como una producion epsilon.

Una gramática libre de contexto puede estar formada por un número finito de producciones.

Un ejemplo de una producción de un lenguaje de programación es:

<INSTRUCCION> --> IF <EXP. BOLEANA> THEN <INSTRUCCION>

Uno de los no-terminales es especificado como el símbolo inicial de la gramática, el cual por default toma la primera posición superior izquierda de la primera producción y a partir de éste símbolo se derivan las secuencias de un lenguaje.

Resumiendo lo anterior, una gramática libre de contexto esta definida por:

- Un conjunto finito de no-terminales
- Un conjunto finito de terminales
- Un conjunto finito de producciones de la forma:

$$\langle A \rangle \longrightarrow \alpha$$

Donde <A> es un no-terminal y α es una secuencia (posiblemente una secuencia nula) de terminales y no-terminales, <A> es llamado el lado izquierdo de la producción y α es llamado el lado derecho.

- Un símbolo inicial, el cual es uno de los no-terminales.

2.- DERIVACIONES

Dentro de ésta seccion se discutirá, como las gramáticas son usadas para generar las secuencias de un lenguaje.

Las reglas o producciones son usadas para definir un modo especial en la sustitucion de una cadena. Esta sustitución se lleva a cabo reemplazando un no-terminal especificado dentro de alguna cadena de terminales y no-terminales que se encuentra más a la izquierda del lado derecho de la produccion.

Algunas veces se dice que la produccion es aplicada al no-terminal en la cadena.

Por ejemplo; considerando la siguiente gramática con el símbolo inicial <S>

1. <S> --> a <A> <B> c
2. <S> --> ε
3. <A> --> c <S> <B>
4. <A> --> <A> b
5. <B> --> b <B>
6. <B> --> a

Dada la siguiente cadena de terminales y no-terminales:

a <A> <B> c
  ↑
  5

Aplicando la producción 5 al no-terminal <B>, apuntado por la flecha (hacia arriba), el resultado de la correspondiente sustitucion es:

a <A> b <B> c

Y así sucesivamente aplicando las siguientes producciones con el orden dado a continuación.

<S> => a <A> <B> c => a <A> b <B> c => a c <S> <B> b <B> c
       ↑   ↑   ↑   ↑
       1   4   3   6

=> a c <S> a b <B> c => a c a b <B> c => a c a b a c
      ↑   ↑
      2   6

Se obtiene como resultado la cadena a c a b a c derivada a partir del símbolo inicial <S>

Las cadenas o cuerdas se definen como una serie de elementos concatenados, la cual solamente esta formada por elementos terminales.

El símbolo ε (epsilon), representa la cadena vacia; cuya concatenacion de la cadena vacia con cualquier cosa es igual a cualquier cosa.

3.- NOTACION UTILIZADA POR LAS GRAMATICAS LIBRES DE CONTEXTO

Para poder diferenciar los terminales de los no-terminales asi como otras representaciones, las siguientes reglas de notacion en las gramaticas libres de contexto son aplicables.

3.1 Simbolos que generalmente son no-terminales

i) Nombres de expresiones, declaraciones, operaciones, etc.

ii) Las primeras letras mayusculas del alfabeto: A,B,C,...;

iii) La letra S, la cual aparece generalmente como el símbolo inicial de una gramática.

3.2 Los símbolos terminales pueden ser representados por:

i) Las primeras letras minusculas del alfabeto; a,b,c,...;

ii) Los operadores tal como +,-,etc.

iii) Símbolos de puntuacion tal como: parentesis; "(",")",",", etc.

iv) Los digitos 1,...,9

3.3 La representación de cadenas de la gramática

La cual puede ser por medio de las primeras letras griegas; α , β , γ , etc.

3.4 Clasificación de los elementos que forman una gramatica libre de contexto.

Considerando la siguiente gramática;

1. S --> B c d
2. S --> A
3. A --> B A c
4. A -->
5. B --> a

Clasificación:

$G = \{\Sigma, N, P, S\}$ Conjunto de cantidades que forman la gramatica libre de contexto.

$\Sigma = \{a, c, d\}$ Alfabeto de la gramática o conjunto de elementos terminales.

$N = \{S, A, B\}$ Conjunto de elementos no-terminales.

S Símbolo inicial de la gramatica, todas las cuerdas se deben generar a partir de este simbolo, con la aplicación de las producciones de la gramática.

$P = \{1,2,3,4,5\}$	Conjunto de producciones.
$\alpha =$ B c d	Representa una cadena de la gramática.
ε (epsilon)	Representa la cuerda o cadena vacía.

NOTA: Se puede observar que la gramática solo contiene no-terminales en el lado izquierdo, lo que la caracteriza como una gramática libre de contexto.

4.- ALGORITMOS

Antes de entrar en detalle en el procedimiento para eliminar los elementos inútiles que pueden aparecer dentro de las gramáticas libres de contexto se darán algunas definiciones.

Elementos inútiles son aquellos que están formados tanto por elementos inaccesibles (terminales y no-terminales) como por elementos muertos (no-terminales).

Aquellos símbolos no-terminales que no son capaces de generar cadenas o cuerdas, se les llama símbolos muertos.

Un símbolo terminal y no-terminal es inaccesible si no se puede llegar a el desde símbolo inicial "S".

5.- ALGORITMO : ELIMINACION DE SIMBOLOS INUTILES

5.1 Eliminación de Elementos muertos

1) Hacer una lista de no-terminales que tengan al menos una producción sin no-terminales en el lado derecho.
2) Si se encuentra una producción tal que todos los no-terminales del lado derecho se encuentran en la lista, aumente a esta el no-terminal del lado izquierdo.
3) Repetir el paso 2), hasta no poder aumentar más elementos, (los elementos no-terminales que no estén en la lista son muertos).

5.2 Eliminación de Elementos Inaccesibles

1) Iniciar una lista con el símbolo inicial de la gramática.
2) Si se encuentra una producción tal que el símbolo del lado izquierdo se encuentra en la lista se añaden a ella todos los símbolos (terminales y no-terminales) del lado derecho.
3) Repetir el paso 2) hasta no, poder aumentar más símbolos, (los elementos que no estén en la lista son inaccesibles).

El procedimiento de éste algoritmo se debe hacer en éste orden , de lo contrario si se eliminan primero los elementos inaccesibles y despues los elementos muertos, se pueden encontrar de nuevo inaccesibles.

Ejemplo: La gramática de entrada es; $G = \{N, \Sigma, P, S\}$

1. S --> A
2. A --> a B
3. A --> a b c
4. B --> A B
5. C --> A d
6. C --> ε

- Eliminando elementos muertos

L : A, C / S

B : es elemento muerto

S --> A
A --> a b c
C --> A d
C --> ε

- Eliminando elementos inaccesibles

L : S / A / a b c

C : es elemento inaccesible

La gramatica equivalente de salida es:

$G' = \{N', \Sigma', P', S\}$; $N' = \{S, A\}$; $\Sigma' = \{a, b, c\}$

P' $\begin{cases} 1.\ S \text{ --> } A \\ 2.\ A \text{ --> } a\ b\ c \end{cases}$

6.- TIPOS DE PRODUCCION

Aqui se hace mención de los diferentes tipos de producción, los cuales serán analizados por los siguientes algoritmos.

1) A --> w : producción del tipo A, un no-terminal produce una secuencia de terminales y no-terminales.
2) A --> ε : producción epsilon, produce la cuerda vacia(no contiene elementos).
3) A --> B : producción sencilla (single production), un no-terminal produce otro no-terminal.

4) A --> A w: producción directamente recursiva por la izquierda, un no-terminal produce el mismo no-terminal del lado izquierdo seguido por cualquier serie de terminales y no-terminales.

7.- ALGORITMO : ELIMINACION DE PRODUCCIONES-EPSILON

Una gramática libre de contexto es $\mathcal{E}$-FREE si y sólo si:
a).- En P(conjunto de producciones), no hay producciones epsilon.
b).- O la única producción epsilon es S --> $\mathcal{E}$ y S no aparece al lado derecho de ninguna producción.
Ejemplo: Supongamos que se tiene la siguiente gramática

1. <DECLARACION> --> $\mathcal{E}$
2. <DECLARACION> --> TIPO <IDENTIFICADOR> <LISTA>
3. <LISTA> --> <LISTA> , <IDENTIFICADOR>
4. <LISTA> --> $\mathcal{E}$

Se puede observar que la gramática no es $\mathcal{E}$-FREE debido a a la producción 4.

PROCEDIMIENTO :

1) Hacer una lista de no-terminales que esten en el lado izquierdo de una producción epsilon.
2) Si se encuentra una producción tal que todos los elementos del lado derecho son no-terminales y estan en lista anterior, anadir a ella el no-terminal del lado izquierdo.
3) Repetir el paso 2) hasta no poder aumentar más elementos a la lista.
4) Por cada producción de la forma A --> $w_0 B_1 w_1 B_2 w_2 \ldots B_K w_K$ en donde B_i es un no-terminal contenido en la lista y $w_a \notin (N \cup \Sigma)^*$ pero ningun no-terminal está en la lista; aumentar todas las producciones A --> $w_0 C_1 w_1 C_2 w_2 \ldots C_K w_K$ que se puedan formar, dando a C_i los valores: $\mathcal{E}$ y B_i
5) Si S(símbolo inicial), esta en la lista anada las producciones; S'--> S y S'--> $\mathcal{E}$

* : indica la cerradura de un lenguaje, la cual se se define como todas las posibles concatenaciones que se puedan formar con las cadenas de un lenguaje.

Ejemplo: Sea la gramatica de entrada $G = \{N, \Sigma, P, S\}$

1. S --> A a B
2. S --> ϵ
3. A --> B C
4. A --> a
5. B --> b A S
6. B --> ϵ
7. C --> a B B c A
8. C --> ϵ

Lista : S, B, C / A

S --> A a B : S --> a B
S --> A a
S --> a

A --> B C : A --> C
A --> B

B --> b A S : B --> b S
B --> b A
B --> b

C --> a B B c A : C --> a B c A
C --> a c A
C --> a B c
C --> a c
C --> a B B c

La gramatica equivalente ϵ-FREE de salida es:

$G' = \{N, \Sigma, P', S'\}$

1. S' --> S
2. S' --> ϵ
3. S --> A a B
4. S --> a B
5. S --> A a
6. S --> a
7. A --> a
8. A --> B C
9. A --> C
10. A --> B
11. B --> b A S
12. B --> b S
13. B --> b A
14. B --> b
15. C --> a B B c A
16. C --> a B c A
17. C --> a c A

18. C --> a B c
19. C --> a c
20. C --> a B B c

8.- ALGORITMO : ELIMINACION DE PRODUCCIONES SENCILLAS

a) Una gramatica libre de ciclos es aquella, en la que no ocurre que $A \stackrel{*}{\Rightarrow} A$ para ningun no-terminal A
b) Para obtener una gramatica libre de ciclos, se debe partir de una gramatica ϵ-FREE

PROCEDIMIENTO :

1) Para cada no-terminal inicializar una lista con ese no-terminal.
2) Si hay una produccion de la forma B --> C y B esta en la lista, anadir a ésa lista el no-terminal C
3) Repetir el paso 2) hasta no poder anadir mas elementos a las listas.
4) Si B --> ω es una producción de la gramática original, anadir a P' todas las producciones de la forma A --> ω (no es produccion sencilla), para todas las A tales que B este en la lista de A.

Ejemplo: La gramatica de entrada es la siguiente;

$G = \{N, \Sigma, P, S\}$

1. S --> p B c
2. S --> A
3. S --> ϵ
4. A --> D p A
5. A --> C
6. B --> b C
7. B --> A
8. C --> a
9. C --> D
10. D --> p B
11. D --> d

L(S) : S A C D

L(A) : A C D

L(B) : B A C D

L(C) : C D

L(D) : D

La gramatica equivalente G' es:

$G' = \{N, \Sigma, P', S\}$

1. S --> p B C
2. S --> ε
3. S --> D p A
4. S --> a
5. S --> p B
6. S --> d
7. A --> D p A
8. A --> a
9. A --> p B
10. A --> d
11. B --> b C
12. B --> D p A
13. B --> a
14. B --> p B
15. B --> d
16. C --> a
17. C --> p B
18. C --> d
19. D --> p B
20. D --> d

9.- ALGORITMO : ELIMINACION DE RECURSION POR LA IZQUIERDA

Para obtener una gramatica no recursiva por la izquierda se debe partir de una gramática propia.
Una gramática libre de contexto, es propia si y sólo si:

- Es libre de elementos inutiles
- Es ε-FREE
- Es libre de ciclos

Sea G una gramática libre de contexto propia con un conjunto $N = \{A_1, A_2, \ldots, A_n\}$

1) Hacer i=1
2) Reemplazar las producciones:

$A_i \longrightarrow A_i \alpha_1 / A_i \alpha_2 / \ldots / A_i \alpha_m / \beta_1 / \beta_2 / \ldots / \beta_p$

por las producciones:

$A_i \longrightarrow \beta_1 / \beta_2 / \ldots / \beta_p / \beta_1 A'_i / \beta_2 A'_i / \ldots / \beta_p A'_i$

$A'_i \longrightarrow \alpha_1 / \alpha_2 / \ldots / \alpha_m / \alpha_1 A'_i / \alpha_2 A'_i / \ldots / \alpha_m A'_i$

NOTA : en la produccion $A_i \rightarrow \beta_j$ para $j=1,2,\ldots,p$ ninguna β_j comienza con una A_k tal que $k \leq i$

3) Si i=n forme G' con $N' = N \cup \{A'_i\}$ y P' = P excepto en las producciones substituidas en el paso 2). En caso cotrario haga i=i+1 ; j=1

4) Reemplace cada produccion de la forma

$A_i \rightarrow A_j \alpha$ por las producciones:

$A_i \rightarrow \beta_1 \alpha \ / \ \beta_2 \alpha \ / \ldots / \ \beta_m \alpha$

en donde $A_j \rightarrow \beta_1 \ / \ \beta_2 \ / \ldots / \ \beta_m$ son todas las; $A_j \rightarrow$ producciones

5) Si j=i-1 pasar a 2) en caso contrario j=j+1 y pasar a 4)

Ejemplo: Sea la gramatica propia de entrada $G = \{N, \Sigma, P, S\}$

1. A --> B C
2. A --> a
3. A --> ε
4. B --> C A
5. B --> A b
6. C --> A B
7. C --> c c
8. C --> a

Donde $N = \{\underset{A_1}{A} \ \underset{A_2}{B} \ \underset{A_3}{C}\}$: n=3

1: i=1

2: A --> $\underbrace{B\ C}_{\beta_1}$

A --> $\underbrace{a}_{\beta_2}$

no contiene producciones de A que comienzan con A

3: i=n no cumple => i=i+1 ; j=1

4: B --> C A ; B --> A b

si contiene producciones de B que comienzan con A

B --> $A\ \underbrace{b}_{\alpha}$

reemplazando las produciones de A :

B --> $\underbrace{B\ C}_{\beta_1}\ \underbrace{b}_{\alpha}$

B --> $\underbrace{a}_{\beta_2}\ \underbrace{b}_{\alpha}$

B --> C A

5: j=i-1 si cumple => regresa a 2)

2: B --> B C b ; B --> C A

B --> a b

si contiene producciones de B que comienzan con B reemplazando:

B --> a b
B --> C A
B --> a b B'
B --> C A B'
B' --> C b
B' --> C b B'

3: i=n no cumple => i=i+1 ; j=1

4: C --> A B

C --> c c
C --> a
si contiene producciones de C que comienzan con A reemplazando las producciones de A
C --> B C B

C --> a B

C --> c c
C --> a

5: j ≠ i - 1 no cumple => j=j+1 ; pasa a 4)

4: C --> B C B

C --> a B
C --> c c
C --> a
si contiene producciones de C que comienzan con B reemplazando las producciones de B(pero no las de B")

B --> a b

B --> C A

B --> a b B'

B --> C A B'

C --> a b C B

C --> C A C B

C --> a b B'C B

C --> C A B'C B

C --> a B

C --> c c

C --> a

5: j=i-1 si cumple => regresa a 2)

2: C --> a b C B
   β_1

C --> C A C B
   α_1

C --> a b B'C B
   β_2

C --> C A B'C B
   α_2

C --> a B
   β_3

C --> c c
   β_4

C --> a
   β_5

si contiene producciones de C que comienzan con C reemplazando las producciones de C

C --> a b C B

C --> a b B'C B

C --> a B

C --> c c

C --> a

C --> a b C B C'

C --> a b B'C B C'

C --> a B C'

C --> c c C'

C --> a C'

C' --> A C B

C' --> A B' C B

C' --> A C B C'

C' --> A B' C B C'

3: i=n si cumple => se forma G' con N' = N U $\{A'\}$ y P' = P
(P' = conjunto de producciones resultantes)

La gramatica equivalente de salida es:

G' = $\{N', \Sigma, P', S\}$

1. A --> B C
2. A --> a
3. A --> ε
4. B --> a b
5. B --> C A
6. B --> a b B'
7. B --> C A B'
8. B' --> C b
9. B' --> C b B'
10. C --> a b C B
11. C --> a b B' C B
12. C --> a B
13. C --> c c
14. C --> a
15. C --> a b C B C'
16. C --> a b B' C B C'
17. C --> a B C'
18. C --> c c C'
19. C --> a C'
20. C' --> A C B
21. C' --> A B' C B
22. C' --> A C B C'
23. C' --> A B' C B C'

10.- FACTORIZACION

El principio de la factorizacion queda establecido en terminos de la siguiente simbologia; una gramatica que contiene n producciones.

<A> --> α β_1
.
.
.
<A> --> α β_m

Donde <A> es un no-terminal, α y β_i para $1 \leq i \leq n$ son secuencias de terminales y no-terminales, entonces las n producciones pueden ser reemplazadas por n+1 producciones

<A> --> α <NUEVO>

<NUEVO> --> β_1
.
.
.
<NUEVO> --> β_m

Donde <NUEVO> es un nuevo no-terminal de la gramática original.

La gramática obtenida una vez aplicado e algoritmo de factorización genera el mismo lenguaje que la gramática original.

La aplicacion del algoritmo de factorización tiene como objetivo principal; eliminar la interseccion existente en los conjuntos de seleccion de gramaticas que quieran ser convertidas en gramaticas del tipo LL(1)

Ejemplo: Suponga la siguiente gramática original formada por dos producciones;

<S> --> IF <B> THEN <S1>
<S> --> IF <B> THEN <S1> ELSE <S>

Esta gramatica no es del tipo LL(1) ya que contiene dos producciones con el mismo lado izquierdo como intersección en sus conjuntos de seleccion IF

Aplicando el algoritmo de factorización para eliminar la interseccion existente entre los conjuntos de selección, podemos obtener una gramática equivalente del tipo LL(1) compuesta por tres producciones (n+1), la cual genera el mismo lenguaje que la gramática original.

Gramática equivalente LL(1) :

<S> --> IF <B> THEN <S1> <NUEVO>
<NUEVO> --> ELSE <S>
<NUEVO> --> ε

OBTENCION DE LA TABLA DE CONTROL

PARSER TOP-DOWN

Todo procedimiento que involucra la operación de reconocimiento de producciones, es llamado "PARSER". Donde la palabra parsing se deriva del gramático de un lenguaje natural, que significa, el arte de hablar y escribir correctamente un lenguaje.

Cabe hacer notar que el método de parser utilizado en la generación de la tabla de control, es el parser Top-Down cuyo reconocimiento es predictivo, indicando con esto que en todo momento el contenido del stack es una predicción de lo que debe haber a la entrada.

A continuación se describen los 13 pasos que comprenden el procedimiento para obtener la tabla de control.

1. ENCONTRAR LOS NO-TERMINALES ANULABLES Y PRODUCCIONES ANULABLES

i) Eliminar todas las producciones que tengan al menos un terminal en el lado derecho.

ii) De la gramática resultante, determinar que no-terminales están muertos y cuales vivos, de acuerdo al procedimiento dado en la sección 5.1 del capítulo II.

iii) Son no-terminales anulables los que resulten vivos en el paso anterior.
Son producciones anulables aquellas que sólo tengan no-terminales anulables en el lado derecho.

Nota : Una producción anulable produce la cuerda vacia $\in$ (epsilon) : es anulable
(vea ejemplo, pag. 24)

2. CONSTRUIR LA RELACION : COMIENZA DIRECTAMENTE CON BDW (BEGINS-DIRECTLY-WITH)

Dos simbolos A y B en una gramatica están relacionados por BDW :

A BEGINS-DIRECTLY-WITH B

Si y sólo si; existe una producción de la forma:
$A \dashrightarrow \alpha\ B\ \beta$ $(A \in N, \alpha \in N^*, B \in (N \cup \in), \beta \in (N \cup \in)^*$

Donde α es anulable y β representa cualquier cadena.
Dada la siguiente producción:

<A> --> <V> <X> <Y> <Z>

De acuerdo a la forma $A \dashrightarrow \alpha B \beta$ se puede observar que α toma diferentes valores:

Para A = <A> , $\alpha = \varepsilon$, B = <V>, β = <X> <V> <Z> , tenemos:
<A> BEGINS-DIRECTLY-WITH <V>
Si se satisface la condicion de que α es anulable.

A = <A> , α = <V>, B = <X>, β = <V> <Z>

<A> BEGINS-DIRECTLY-WITH <X>

Si <X> es anulable

A = <A>, α = <V> <X>, B = <V>, β = <Z>

<A> BEGINS-DIRECTLY-WITH <Y>

Si <Y> es anulable

<A> BEGINS-DIRECTLY-WITH <Z>

Estas relaciones pueden ser representadas por medio de una tabla o matriz, donde los renglones y columnas son etiquetadas con los simbolos terminales y no-terminales de la gramática analizada y cada relacion obtenida es marcada con "1" en la posicion que le corresponde en la matriz.
(vea ejemplo, pag. 25)

3. CONTRUIR LA RELACION : COMIENZA CON BW(BEGINS-WITH)

Si A y B son simbolos de una gramatica y están relacionados por BW

A BEGINS-WITH B

Si y solo si existe una cadena que empiece con B y pueda ser derivada de A o sea que las posisiones marcadas con "1" en el renglon B son copiadas al renglon A
La relacion BW(BEGINS-WITH) se define como la cerradura transitiva reflexiva de la relacion BDW(BEGINS-DIRECTLY-WITH):

$BW = BDW^*$

Donde tal relación se compone de todas las relaciones R obtenidas a partir de los pares(a,b) de elementos ordenados de un conjunto llamado S, tal que a=b; y los simbols "*" (astériscos), asignados a las posiciones de la diagonal principal de la matriz representan los pares anadidos a la relación BEGINS-DIRECTLY-WITH por el consepto de cerradura transitiva reflexiva.

* : Indica todas las posibles composiciones de una relación consigo misma.
(vea ejemplo, pag. 25)

4. ENCONTRAR EL FIRST-SET DE CADA NO-TERMINAL

El conjunto FIRST (<A>) es simplemente un conjunto de terminales b tal que ;

<A> BEGINS-WITH b

FIRST <A> = $\{b/b \in \Sigma$ y BW (A,b) = 1$\}$

El conjunto FIRST para cada no-terminal <A> se forma con los terminales que hayan sido marcados con "1" en la tabla BEGINS-WITH.(vea ejemplo, pag. 25)

5. ENCONTRAR EL FIRST-SET DE CADA PRODUCCION

El FIRST para el lado derecho de cada producción se obtiene facilmente, aplicando las siguientes reglas para efectuar la unión de conjuntos dentro de las gramáticas libres de contexto, considerando los conjuntos (FIRST-SET) formados en el paso 4.

Si A --> ω es una producción, entonces :

i) FIRST(ω) = ϕ si $\omega = \varepsilon$: ϕ = conjunto vacio

ii) FIRST(ω) = $\{a\}$ si ω = a

iii) FIRST(ω) = FIRST(ω) si $\omega \in N$

iv) Si $\omega = X_1 X_2 X_3 \ldots X_k$ k>1

FIRST(ω) $\bigcup_{i}^{m}$ FIRST (X_i) M $\leq$ k

y X_m es el primer elemento no-anulable en ω

(vea ejemplo, pag. 25)

6. CONSTRUIR LA RELACION : ESTA SEGUIDO DIRECTAMENTE POR IFDB(IS-FOLLOWED-DIRECTLY-BY)

Para el lado derecho de cada producción.
Dos símbolos <A> y <B> estan relacionados por IFDB

<A> IS-FOLLOWED-DIRECTLY-BY <B>

Si existe una producción de la forma:

C --> α A β B γ
Donde A,B $\in N \cup \Sigma$, α , $\gamma \in (N \cup \Sigma)^*$ y $\beta \in N^*$ A y B son símbolos, β es una cadena anulable, α y γ son cadenas arbitrarias. (vea ejemplo, pag. 26)

7. CONTRUIR LA RELACION : ES DIRECTAMENTE EL FINAL DE IDEO(IS-DIRECTLY-END-OF)

Para una produción A --> w y cualquier elemento B contenido en w se tiene que:

B IS-DIRECTLY-END-OF A

Si y sólo si todos los elementos a la derecha de B son anulabes. (vea ejemplo, pag. 26)

8. CONTRUIR LA RELACION : ES EL FINAL DE IEO(IS-END-OF)

Dos simbolos A y B estan relacionados por:

A IS-END-OF B

Si y sólo si : B $\stackrel{*}{\Rightarrow}$ w y w termina con A aplicando 0,1 o'más producciones

IEO = IDEO*

Una vez obtenida la relación IDEO,IEO se procede con la aplicacion del cosepto de cerradura transitiva reflexiva.
(vea ejemplo, pag. 26)

9. CONSTRUIR LA RELACION : ESTA SEGUIDO POR IFB(IS-FOLLOWED-BY)

Esta relacion también es llamada PRODUCTO DE RELACIONES.
Dados los simbolos A,B,X,Y de una gramatica se tiene que:

<A> IS-FOLLOWED-BY <B>

Si y solo si:

$S \stackrel{*}{\Rightarrow} w_0$ X Y $w_1 \stackrel{*}{\Rightarrow} w_2$ A B w_3 tal que:

<A> IS-END-OF <X> IS-FOLLOWED-DIRECTLY-BY <Y> BEGINS-WITH <B>

Lo que se equivalente a:

IFB = IEO.IFDB.BW

(vea ejemplo, pag. 27,28 y 29)

10. AMPLIAR LA RELACION IFB PARA INCLUIR EL SIMBOLO " ⊣ " (FIN DE CUERDA)

i) Se aumenta una columna a IFB
ii) Se copia en ella la columna del símbolo inicial "S" de la relacion IEO
(vea ejemplo, pag. 29)

11. CALCULAR EL FOLLOW-SET PARA CADA NO-TERMINAL ANULABLE

Sea <A> un no-terminal anulable

$$FOLLOW(<A>) = \{ b/b \ IFB A,\ b \in \Sigma \cup \{\dashv\} \}$$

El FOLLOW-SET se define como el conjunto de simbolos $b \in \Sigma \cup \{\dashv\}$ que pueden aparecer a continuacion de X en una forma sentencial derivada de S ⊣

El FOLOW-SET de cada no-terminal anulable se forma unicamente con los simbolos terminales marcados en la tabla ampliada IFB. (vea ejemplo pa. 29)

12. CONTRUIR EL SELECTION-SET PARA CADA PRODUCCION

El conjunto de seleccion para cada produccion(i): A --> α
Se define como:

$$s.s.(i) = \begin{cases} FIRST(\alpha) & \text{si } \alpha \text{ no es anulable} \\ FIRST(\alpha) \cup FOLLOW(A) & \text{si } \alpha \text{ es anulable} \end{cases}$$

Una vez obtenidos los conjuntos de seleccion para cada produccion, se podra determinar si la gramatica analizada es del tipo LL(1).

Una gramatica libre de contexto se dice que es LL(1) si y sólo si; los conjuntos de seleccion para dos producciones con el mismo lado izquierdo son disjuntos, o sea que no existe interseccion entre ellos.
(vea ejemplo, pag. 30)

13. DISENO DE LA TABLA DE CONTROL PARA UN PARSER TOP-DOWN PARA PARA RECONOCER EL LENGUAJE DE UNA GRAMATICA LL(1)

13.1 Características del parser Top-Down

1) El alfabeto de entrada del autómata, es el mismo que el de la gramática mas {—|}"fin de cuerda"
2) El stake inicial es: ▽S
3) El autómata tiene un sólo estado {q_0}
4) El alfabeto del stack {Γ} es:
{▽} U N U {terminales que aparecen en otra posición que no sea el inicio de una producción}
5) Tabla de control:

Es una forma de representar al autómata de un sólo estado, la cual está formada por renglones etiquetados con los símbolos del stack, columnas etiquetadas con los símbolos de entrada, más las entradas que se describen a continuación.

Función de Tránsición {δ} = ME [A,b]
b = símbolo de entrada
A = símbolo del stack

6) Cada producción de la gramática corresponde a a una entrada en al tabla de control. Si existe una producción de la forma:
A --> b α
Donde: A es un no-terminal
b es un terminal
y es una cadena de terminales y no-terminales.
Donde la entrada correspondiente a ésta producción es el renglon A y la columna b, o sea;

$$M[A,b] = \begin{cases} \text{REEMPLAZA}(\alpha^r) \\ \text{AVANZA} \end{cases}$$

El contenido de la cadena es reemplazado en modo inverso (α^r) en el stack.
Si la producción es de la forma A --> b y $\alpha = \epsilon$ entonces;

$$M[A,b] = \begin{cases} \text{POP} \\ \text{AVANZA} \end{cases}$$

7) Si el terminal b es un símbolo del stack, entonces la entrada a la tabla es por el renglón b y la columna b;

$$M[b,b] = \begin{cases} POP \\ AVANZA \end{cases}$$

$\forall\ b \in \Gamma$ y $b \in \Sigma$

8) Si se tiene una producción de la forma: i: A --> ε se tiene que;

$$M[A, s.s.(i)] = \begin{cases} POP \\ RETIENE \end{cases}$$

Al generar la tabla de control es importante recordar que los conjuntos de selección se deben asociar con su respectiva producción.

9) Si existe una producción i: A --> β y β comienza con un no-terminal

$$M[A, s.s.(i)] = \begin{cases} REP(\beta^r) \\ RETIENE \end{cases}$$

El contenido de la cadena β es reemplazado en modo inverso(β^r) en la posición que le corresponde dentro del stack.

10) Si la entrada a la tabla es la posición que corresponde a la configuración final;

$M[\nabla, \dashv]$ = ACEPTA (se acepta la cadena)

11) Todas las posiciones no cubiertas en los puntos 6), 7), 8), 9) y 10) producen "ERROR"
(vea ejemplo, pag. 30 y 31)

Ejemplo: Sea G = {N, Σ , P, S} la gramatica de entrada, obtener sus respectivos conjuntos de selección(selection set) y si se trata de una gramatica del tipo LL(1), proceder a obtener su respectiva tabla de control.

1. S --> a A B
2. S --> ε
3. A --> b S A
4. A --> c c c
5. A --> d
6. B --> A a B
7. B --> a

PASO 1) Encontrar los no-terminales anulables y producciones anulables.

S : elemento anulable

S --> ε : produccion anulable

PASO 2) Construir la relacion BDW(BEGINS-DIRECTLY-WITH)

PASO 3) Construir la relacion BW(BEGINS-WITH)

	S	A	B	a	b	c	d
S	*			1			
A		*			1	1	1
B		*	*	1	*	*	*
a				*			
b					*		
c						*	
d							*

PASO 4) Construir el FIRST-SET para cada no-terminal

FIRST(S) = {a}
FIRST(A) = {b c d}
FIRST(B) = {a b c d}

PASO 5) Construir el FIRST-SET de cada producción

FIRST(a A B) = {a}
FIRST(ε) = ϕ
FIRST(b S A) = {b}
FIRST(c c c) = {c}
FIRST(d) = {d}
FIRST(A a B) = {b c d}
FIRST(a) = {a}

PASO 6) Construir la relacion IFDB(IF-FOLLOWED-DIRECTLY-BY)
(del lado derecho de cada produccion)

	S	A	B	a	b	c	d
S		1					
A				1			
B							
a		1					
b	1	1					
c						1	
d							

PASO 7) Construir la relacion IDEO(IS-DIRECTLY-END-OF)

PASO 8) Construir la relacion IEO(IS-END-OF)

	S	A	B	a	b	c	d
S	*						
A		1					
B	1		1				
a			1	*			
b					*		
c		1				*	
d		1					*

PASO 9) Construir la relación IFB(IS-FOLLOWED-BY)
(o producto de relaciones IFB=IEO.IFDB.BW)

IEO

	S	A	B	a	b	c	d
S	1	0	0	0	0	0	0
A	0	1	0	0	0	0	0
B	1	0	1	0	0	0	0
a	1	0	1	1	0	0	0
b	0	0	0	0	1	0	0
c	0	1	0	0	0	1	0
d	0	1	0	0	0	0	1

.

IFDB

	S	A	B	a	b	c	d
S	0	1	0	0	0	0	0
A	0	0	0	1	0	0	0
B	0	0	0	0	0	0	0
a	0	1	0	0	0	0	0
b	1	1	0	0	0	0	0
c	0	0	0	0	0	1	0
d	0	0	0	0	0	0	0

=

IED.IFDB

	S	A	B	a	b	c	d
S	0	1	0	0	0	0	0
A	0	0	0	1	0	0	0
B	0	1	0	0	0	0	0
a	0	1	0	0	0	0	0
b	1	1	0	0	0	0	0
c	0	0	0	1	0	1	0
d	0	0	0	0	1	0	0

BW

	S	A	B	a	b	c	d
S	1	0	0	1	0	0	0
A	0	1	0	0	1	1	1
B	0	1	1	1	1	1	1
a	0	0	0	0	0	0	0
b	0	0	0	0	1	0	0
c	0	0	0	0	0	1	0
d	0	0	0	0	0	0	1

IFB

	S	A	B	a	b	c	d
S	0	1	0	0	1	1	1
A	0	0	0	1	0	0	0
B	0	1	0	0	1	1	1
a	0	1	0	0	1	1	1
b	1	1	0	1	1	1	1
c	0	0	0	1	0	1	0
d	0	0	0	1	0	0	0

PASO 10) Ampliar la relacion IFB para incluir el símbolo: " $\dashv$ " (fin de cuerda)

	S	A	B	a	b	c	d	$\dashv$
S	0	1	0	0	1	1	1	1
A	0	0	0	1	0	0	0	0
B	0	1	0	0	1	1	1	1
a	0	1	0	0	1	1	1	1
b	1	1	0	1	1	1	1	0
c	0	0	0	0	0	1	0	0
d	0	0	0	0	0	0	0	0

PASO 11) Calcular el FOLLOW-SET para cada no-terminal anulable; como S es el unico elemento anulable.

FOLLOW(S) = $\{b\ c\ d\ \dashv\}$

PASO 12) Construir el SELECTION-SET para cada produccion.

s.s.(1) = {a}
s.s.(2) = ∅ U FOLLOW(S) = {∅ U b c d ⊣}
s.s.(3) = {b}
s.s.(4) = {c}
s.s.(5) = {d}
s.s.(6) = FIRST(A) = {b c d}
s.s.(7) = {ε}

Una vez obtenidos los conjuntos de seleccion para cada produccion, se determina si no existe interseccion entre los conjuntos de seleccion de producciones con el mismo lado izquierdo, si se cumple la condicion, entonces la gramatica si el del tipo LL(1) y por lo tanto se procede a obtener la tabla de control.

PASO 13) Generacion de la tabla de control.
La tabla es generada a partir del reconocimiento de de cada tipo de produccion y asociando los conjuntos de seleccion con su respectiva produccion.

	a	b	c	d	⊣
S	REP(BA) AVAN	POP RET	POP RET	POP RET	POP RET
A	ERROR	REP(aS) AVAN	REP(cc) AVAN	POP AVAN	ERROR
B	POP AVAN	REP(BaA) RET	REP(BaA) RET	REP(BaA) RET	ERROR
a	POP AVAN	ERROR	ERROR	ERROR	ERROR
c	ERROR	ERROR	POP AVAN	ERROR	ERROR
∇	ERROR	ERROR	ERROR	ERROR	ACEPTA

13.1 Probar si las siguientes cadenas son aceptadas por el autómata Top-Down.

Las cuerdas o cadenas que a continuación se dan son generadas a partir de las producciones de la gramatica de entrada, aplicando "Derivacion por la Izquierda", que consiste en sustituir el símbolo no-terminal que se encuentre mas a la izquierda, (este concepto de derivacion se explica mas ampliamente en la seccion 2. del capitulo II).

Dadas las siguientes cuerdas:

1) a c c c b d a a ⊣
2) a c c ⊣
3) a b d a ⊣

Mostrar los mivimientos realizados por el autómata Top-Down, para cada una de las cadenas, y determinar si estas son aceptadas o rechazadas por el autómata.

Probando la cadena 1)

(∇ S	a c c c b d a a ⊣)
(∇ B A	c c c b d a a ⊣)
(∇ B c c	c c b d a a ⊣)
(∇ B c	c b d a a ⊣)
(∇ B	b d a a ⊣)
(∇ B a A	d a a ⊣)
(∇ B a	a a ⊣)
(∇ B	a ⊣)
(∇ (	⊣)

Una vez efectuados todos los movimientos, se determina si la configuración obtenida es la que pertenece al autómata.
Respuesta:
SE ACEPTA LA CADENA

Probando la cadena 2)

(∇ S	a c c ⊣)
(∇ B A	c c ⊣)
(∇ B c c	c ⊣)
(∇ B c	⊣)

Respuesta:
SE RECHAZA LA CADENA

Probando la cadena 3)

(∇ S	a b d a ⊣)
(∇ B A	b d a ⊣)
(∇ B A S	d a ⊣)
(∇ B A	d a ⊣)
(∇ B	a ⊣)
(∇	⊣)

Respuesta:
SE ACEPTA LA CADENA

RESULTADOS Y CONCLUSIONES

1. CARACTERISTICAS PRINCIPALES DEL EQUIPO DE COMPUTACION UTILIZADO EN EL DESARROLLO DEL SISTEMA MANIPULADOR DE GRAMATICAS.

1.1 Equipo de computacion
El equipo de computacion con el cual se conto para la implantacion del sistema manipulador de gramaticas, fué un microcomputador marca APPLE II-E

1.2 Lenguaje de programacion
El lenguaje utilizado para la implantacion del sistema fue el lenguaje BASIC II-PLUS que es con el cual cuenta la maquina.

1.3 Memoria Interna o principal
La microcomputadora tiene una capacidad de memoria interna(RAM) de 64 Kbytes, de los cuales 29 kbytes son absorbidos por el Sistema Operativo Dos(Disk Operating System), quedando un espacio de 35 Kbytes de memoria principal para el usuario.

1.4 Memoria Externa o Secundaria
Posee dos unidades de disco flexible(para diskette); DRIVE #1 y DRIVE #2, teniendo una capacidad de almacenamiento de 144 Kbytes por diskette.

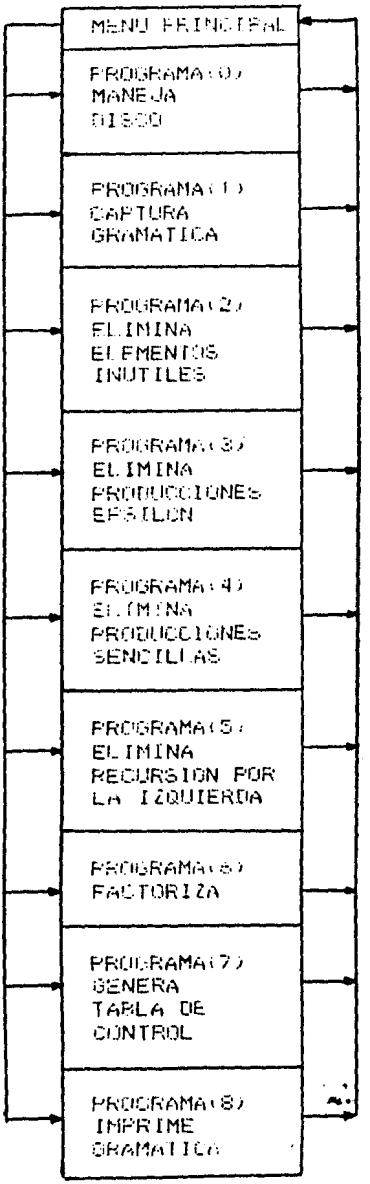
2. ESTRUCTURA DEL SISTEMA

La estructura que define el sistema de información, permite ejecutar ya sea en forma selectiva o en forma secuencial cualquiera de las funciones que integran el sistema.

El tener el mayor número de funciones integradas dentro del sistema, produce como resultado que el sistema sea mucho más rapido en su tiempo de proceso ya que al momento de que el sistema se inicializado es cargado el programa principal(que contiene casi el total de funciones)en memoria principal una sola vez, y de otra manera aunque las aplicaciones son más faciles de implementar usando una serie de dos o mas programas; estos ocupan mas tiempo de proceso, ya que tienen que ser cargados y corridos secuencialmente, aunque tienen la ventaja de que ocupan menos espacio en memoria principal. Para nuestro caso el único programa que corre ligado al programa principal es el que genera la tabla de control debido al número de arreglos o tablas que utiliza.

A continuación se muestra el esquema que define la estructura del sistema.

ESTRUCTURA DEL SISTEMA



3. DEFINICION DE LA ESTRUCTURA DE DATOS UTILIZADA POR EL SISTEMA

Considerando que el sistema requiere del manejo de una gramatica de entrada y una de salida y que ademas sus producciones estan formadas por elementos terminales y no-terminales, se pensó en una representacion interna de las gramaticas, para que estas fuesen manipuladas más facilmente por el sistema y a la vez ocuparan menos espacio en memoria principal y secundaria. Siendo esta representacion interna una matriz de entrada y una de salida, y los elementos terminales y no-terminales que forman las producciones de la gramatica, fueran almacenados en una lista lineal.

Los renglones de la matriz representan las producciones de de la gramatica y las columnas representan tanto, elementos terminales como no-terminales.

Cada elemento terminal y no-terminal, puede ser representado por uno o mas caracteres.

Una lista lineal es un conjunto de nodos; X[1], X[2], ..., X[N]; donde M > 0, cuya propiedad estructural esencial involucra la posicion lineal relativa de los N nodos(en una alineacion). Para una lista lineal se dan las siguientes caracteristicas:

i) Si M>0, X[1] es primer nodo.
ii) Cuando 1<K<M, el nodo X[K] es precedido por X[K-1] y seguido por X[K+1]
iii) X[N] es el último nodo.

Este tipo de lista es utilizado para almacenar los elementos terminales y no-terminales los cuales son clasificados dentro de la lista por medio de dos apuntadores(un apuntador que clasifica a los terminales y otro a los no-terminales).

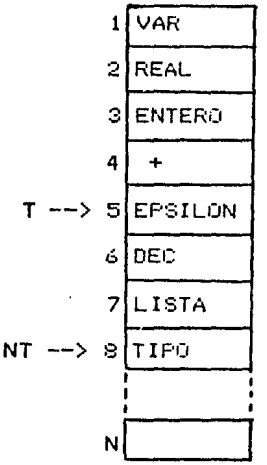
Es importante hacer notar que el apuntador que clasifica a los terminales, siempre apunta a la posicion del último elemento terminal llamado "epsilon" que para nuestra aplicacion se considero dentro de los terminales. Una vez que los elementos han sido introducidos en la lista se procede a formar su respectiva matriz de producciones donde cada produccion es validada de acuerdo a los elementos almacenados en la lista, formando su respectiva produccion o renglon de posiciones las cuales corresponden a las contenidas en la lista.

Para dar una idea más clara sobre la representacion interna, se da el siguiente ejemplo:

1. DEC --> TIPO VAR LISTA
2. LISTA --> + VAR LISTA
3. LISTA --> EPSILON
4. TIPO --> REAL
5. TIPO --> ENTERO

Terminales: VAR,LISTA,+,REAL,ENTERO,EPSILON
No-terminales: DEC,LISTA,TIPO

LISTA LINEAL



MATRIZ

	0	1	2	3	4	5	6		M
1	3	6	8	1	7				
2	3	7	4	1	7				
3	1	7	5						
4	1	8	2						
5	1	8	3						
6									
7									
8									
N									

Significado de la representacion.

LISTA LINEAL : Almacena los elementos terminales y no-terminales.

APUNTADORES : El apuntador T direcciona la posicion del ultimo terminal introducido(epsilon).
NI direcciona la posicion del ultimo no-terminal en la lista.

TABLA o MATRIZ : La columna(i,0) de la matriz almacena el numero de elementos terminales y no-terminales del lado derecho de cada produccion o reglon.
La columna(i,1) almacena el lado izquierdo de cada produccion, siendo siempre este elemento un no-terminal.
Los renglones(i,j) a partir de i=2; almacena todos los elementos terminales y y no-terminales del lado derecho de cada produccion.

4. MANEJO DE ARCHIVOS DE TIPO INDEXADO (ACCESO DIRECTO)

El diseno de la estructura de datos esta basada fundamentalmente en el manejo de archivos de acceso aleatorio(Random Data File); antes de entrar en detalle en su manejo, se dan las definiciones de los principales elementos que intervienen en la estructura de datos.

Un archivo se define como una coleccion de datos con caracteristicas comunes.

Al elemento unitario de un archivo(mínima unidad direccionable)se le llama registro, y a los elementos que componen un registro se les llama campos.

5. CARACTERISTICAS PRINCIPALES DE LOS ARCHIVOS DE TIPO INDEXADO

Mientras que un archivo secuencial de datos, contiene conjuntos de datos que estan ordenados por número de registro creciente, un archivo de datos de acceso aleatorio o acceso directo, consiste en items de datos individuales; que no necesariamente siguen un orden en particular, indicando con ésto, que se puede leer o escribir directamente en un rachivo de datos de acceso aleatorio, sin procesar secuencialmente a lo largo del archivo de datos desde el principio; por tanto es más rápido transferir informacion desde o hacia un archivo de datos de acceso aleatorio que un archivo secuencial de datos.

5.1 Lectura de un Archivo Indexado

Si se van a leer los items de un archivo aleatorio de datos, entonces es necesario considerar la posición del apuntador,ya que este por medio de la instrucción

READ(lectura); debe ser movido hasta la posición adecuada antes de leer cada item de datos.

5.2 Escritura de un Archivo Indexado

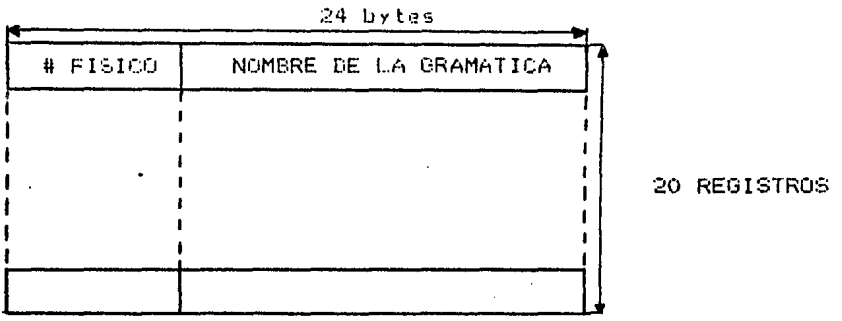
Se puede escribir un item de datos en un archivo aleatorio en forma muy similar como se lee,excepto que ahora se utiliza la proposicion para manejo de archivos WRITE(escritura), como se menciono anteriormente se debe posicionar el apuntador en la localizacion apropiada antes de escribir los items, y el nuevo item de datos reeplazará cualquier informacion almacenada previamente en esa localizacion.

Es importante mencionar que dentro del diseno de la estructura de datos se hace uso del manejo de archivos de nombre "variable" lo cual se explica a detalle en la parte de evaluacion y diseno de la estructura de datos.

6. EVALUACION Y DISENO DE LA ESTRUCTURA DE DATOS UTILIZADA POR EL SISTEMA

Considerando la capacidad de almacenamiento de cada diskette de 144 Kbyte(130,000 caracteres ya formateado el disco), para el diseno de la estructura de datos, y el máximo número de entradas(105), para nombres de archivos diferentes que permite cada directorio del disco asignado para datos; DRIVE #2 . Y el DRIVE #1 asignado para el almacenamiento de los programas de aplicacion. De esto se determino que el siguiente diseno sería el mas optimo.

Un archivo principal en el cual se guardan los nombres de gramaticas



Longitud de cada campo

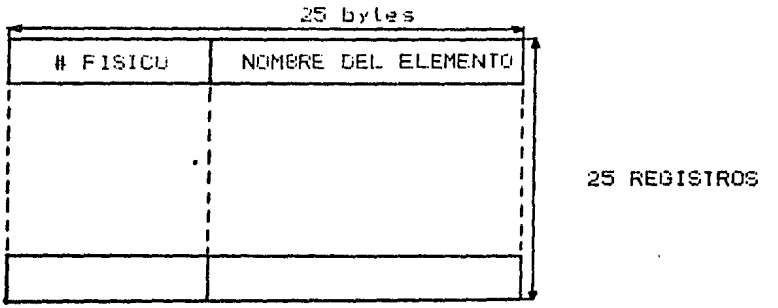
- # FISICO : Está implicito
- NOMBRE DE LA GRAMATICA : 24 caracteres

  Para un máximo de : 20 gramáticas

Asociado a este archivo se encuentran tres archivos más. El primero cuyo nombre se forma de la siguiente manera:

PREFIJO : LETN (lista de elementos terminales y no-terminales).
SUFIJO : X (número físico correspondiente a la localización relativa dentro archivo principal).

Este archivo tiene la siguiente estructura:



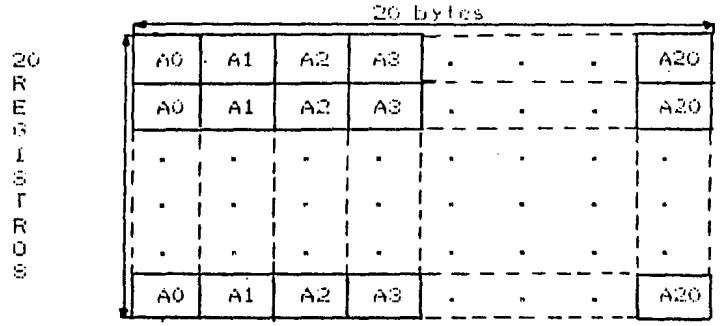
Longitud de cada campo
- # FISICO : Está implícito
- NOMBRE DEL ELEMENTO : 25 caracteres
  Para un máximo de : 25 elementos

El segundo y tercer archivo, cuyos nombres se forman de la siguiente manera:

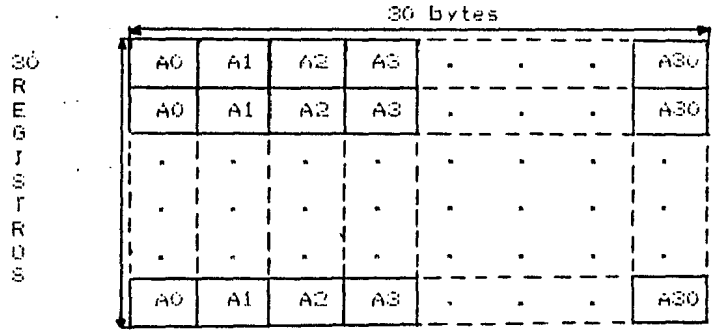
SUFIJO : ME (Matriz de Entrada)
SUFIJO : MS (Matriz de Salida)
PREFIJO : X (Número físico correspondiente a la localización relativa dentro del archivo principal)

Estos dos archivos tienen la misma estructura, la cual se muestra a continuación:

MATRIZ DE ENTRADA "ME"



MATRIZ DE SALIDA "MS"



Donde:

A0 = numero de elementos del lado izquierdo
A1 = elemento izquierdo
A2...N = elementos del lado derecho de c/produc.

Teniendo como máximo para:

Gramatica de Entrada: "ME"
- 20 producciones
- 20 elementos x produc.

Gramatica de Salida: "MS"
- 30 producciones
- 30 elementos x produc.

7. EL SISTEMA SU DESCRIPCION Y USO

El sistema esta integrado por un conjunto de programas los cuales pueden ser manipulados por medio de un número de funciones, el cual se describe a continuacion:

Menu General

0) Maneja Disco

1) Captura Gramatica

2) Elimina Elementos Inutiles

3) Elimina Producciones Epsilon

4) Elimina Producciones Sencillas

5) Elimina Recursión por la Izquierda

6) Factoriza

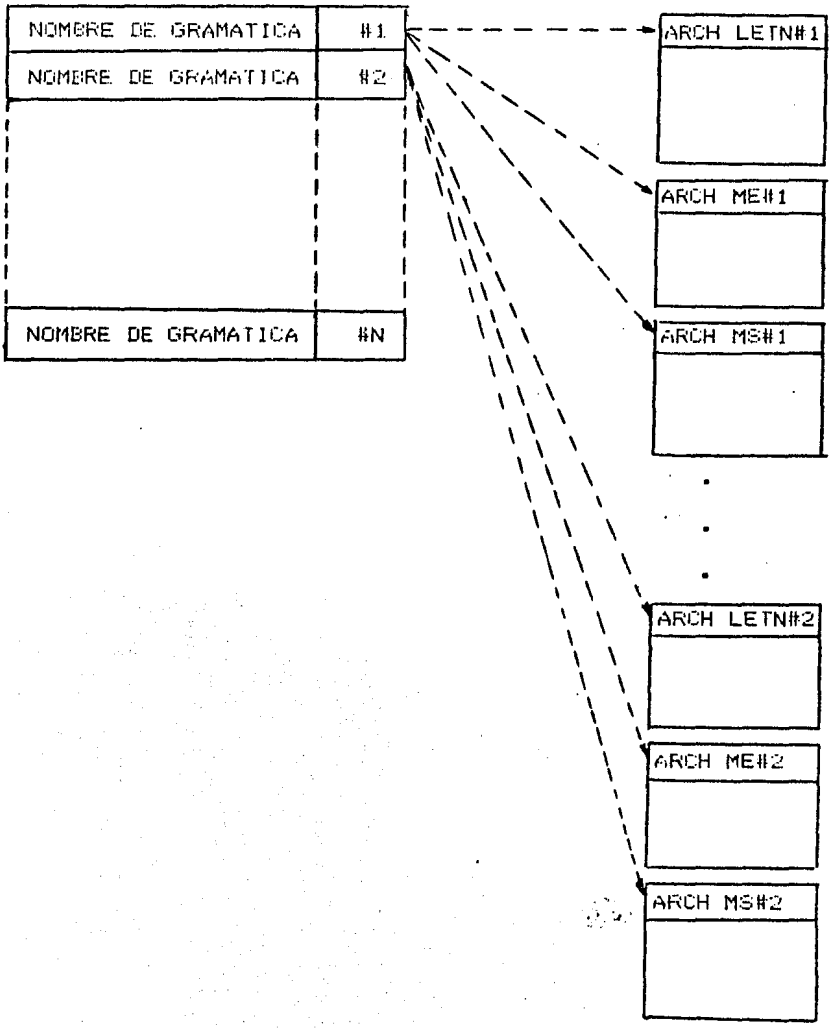
7) Genera Tabla de Control

8) Imprime Gramática

9) Fin

Al inicializarse el sistema, automaticamente un programa monitor carga en memoria principal todos los nombres de las gramaticas grabadas en disco, las cuales han sido capturadas, mediante la función "Captura Gramática", esta función valida la gramática que esta siendo capturada y a su vez produce su representacion interna por medio de una matriz. La gramatica equivalente o de salida tambien es representada por medio de una matriz.

ESQUEMA DE LA ESTRUCTURA DE DATOS DEL SISTEMA



Esta gramática de entrada, una vez procesada puede ser vuelta a su forma original por medio de a función "Imprime Gramática", la cual maneja un submenú con el cual puede presentar, ya sea por pantalla o por impresor, la gramática de entrada, la gramática de salida o ambas.

Otra función importante es la que realiza el "Manejo de Disco", la cual presenta el siguiente submenú de opciones:

1) Graba Gramática
2) Carga Gramática
3) Borra Gramática
4) Catalogo de Gramáticas
5) Fin

Esta función permite al usuario, una vez que la gramática ha sido capturada, poder grabar en disco o procesarla en modo inmediato, seleccionando cualquiera de las funciones proporcionadas por el menu principal o bién, grabarla en disco asignandole un nombre especifico.

Al grabarse una nueva gramática, automaticamente se crean tres archivos de datos con nombre variable, cuyos nombres están relacionados con el nuevo nombre de la gramática, contenido en el archivo principal por medio de un sufijo(número físico correspondiente a la localización relativa dentro del archivo principal).

Las funciones de la 2) a la 6) presentadas en el menú principal pueden ser ejecutadas considerando los requisitos de la siguiente matriz de precedencia(donde las posiciones sin marca, indican que no es requisito).

ES REQUISITO PARA		A	B	C	D	E
ELIMINA ELEMENTOS INUTILES	A				1	
ELIMINA PRODUCC. EPSILON	B			1	1	
ELIMINA PRODUCC. SENCILLAS	C				1	
ELIMINA RECURSION POR LA IZQ.	D					
FACTORIZAC.	E					

La función "Genera Tabla de Control", se encuentra encadenada al programa principal compuesto por las opciones 0), 1), 2), 3), 4), 5), 6) y 8) del menú principal, al momento de ser seleccionada la opción 7) es ejecutado secuencialmente, una vez que la gramática a ser procesada ha sido cargada en memoria principal.

Esta función procesa 12 pasos, los cuales obtienen los conjuntos de selección de la gramática analizada y a partir de estos conjuntos obtenidos se determina si la gramática de entrada es del tipo LL(1), si este es el caso, se procede a obtener su respectiva tabla de control comprendida en el paso 12), en el cual primero se analiza el tipo de producción llenandose a su vez la tabla de control, considerando los conjuntos de selección asociados a cada producción. Ya generada la tabla de control, el sistema pide la cadena a ser analizada (la cual deberá ser dada directamente por el usuario por medio del teclado). Siendo esta parte la que corresponde al simulador del parser top-down.

Durante el análisis de la cadenas; entra en función la utilización de un stack, el cual interactuando con las rutinas de acción, se determina si la cadena analizada es aceptada por el autómata.

7.1 USO DEL SISTEMA

La idea de llevar a cabo esta tesis surgió de la necesidad de tener un sistema interactivo manipulador de gramáticas, que fuese utilizado en forma práctica por los alumnos de la Facultad de Ingeniería de la U.N.A.M. Y de aquí se decidió desarrollar el sistema, el cual una vez puesto en marcha cumplirá con su objetivo.

8. LISTADO DEL SISTEMA Y ALGUNOS EJEMPLOS CON DATOS REALES

En este punto se proporciona el listado completo del sistema y algunos ejemplos, los cuales fueron procesados en el microcomputador(APPLE II-E), en el cual se desarrollo el sistema.

```
LIST

10  REM  **********
20  REM  CABEZA DEL SISTEMA
30  DIM P(20,20),P1(30,30),E$(25),M$(10),L1(20),G(.20),PA(30,30)
40 M$(0) = "MANEJA DISCO"
50 M$(1) = "CAPTURA GRAMATICA"
60 M$(2) = "ELIMINA ELEMENTOS INUTILES"
70 M$(3) = "ELIMINA PRODUCCIONES-E"
80 M$(4) = "ELIMINA PRODUCCIONES SENCILLAS"
90 M$(5) = "ELIMINA RECURSION POR LA IZQUIERDA"
100 M$(6) = "FACTORIZA"
110 M$(7) = "GENERA TABLA DE CONTROL"
120 M$(8) = "IMPRIME GRAMATICA"
130 M$(9) = "FIN"
140 D$ =  CHR$ (4)
150  PRINT D$;"OPEN A-GRAMATICAS,L24,D2,S6": FOR I = 0 TO 20: PRINT D$;"READ A-GRAMATICAS,R";I: INPUT G$(I): NEXT
     I: PRINT D$;"CLOSE A-GRAMATICAS
160  PRINT D$;"ELOAD CHAIN,A520,S1"
170  CALL 520"P-PRINCIPAL,D1,S6"
```

```
LIST

10 D$ = CHR$ (4)
20  PRINT D$;"OPEN A-GRAMATICAS,L24,B2,E6"
30  PRINT D$;"WRITE A-GRAMATICAS,R0": PRINT 1
40  PRINT D$;"WRITE A-GRAMATICAS,R1": PRINT "**********"
50  FOR I = 1 TO 20: PRINT D$;"WRITE A-GRAMATICAS,R";I: PRINT "**********": NEXT : PRINT D$;"CLOSE A-GRAM
    ATICAS"
```

```
LIST

10  REM

20  REM GRAMATICAS LIBRES
30  REM          DE
40  REM       CONTEXTO

50  HOME
60 M$ = "GRAMATICAS LIBRES DE CONTEXTO"
70 B$ = "
80  INVERSE : VTAB 2: HTAB (20 -  LEN (M$) / 2): PRINT M$: NORMAL
90  VTAB 6:T9 = 20: REM # MAXIMO DE GRAMATICAS
100  FOR I = 0 TO 9
110  VTAB I + 6: HTAB 4: PRINT I;" ";M$(I)
120  NEXT I
130  VTAB 20: CALL  - 958: PRINT "CUAL ";: GET R$
140  FOR I = 0 TO 9: IF R$ =  STR$ (I) GOTO 160
150  NEXT I: GOTO 50
160  VTAB 6 + I: HTAB 4: FLASH : PRINT I;" ";M$(I): VTAB 21:A$ = "ESPERA":L =  LEN (M$) / 2: HTAB 20 - L: PRINT M$
     : NORMAL
170  IF R$ = "0" GOTO 290
180  IF R$ = "1" GOTO 1330
190  IF R$ = "3" GOTO 2210
200  IF R$ = "4" GOTO 3250
210  IF R$ = "2" GOTO 1770
220  IF R$ = "5" GOTO 4230
230  IF R$ = "6" GOTO 3480
240  IF R$ = "7" GOTO 5250
250  IF R$ = "8" GOTO 4800
260  IF R$ = "9" GOTO 5240
270  GOTO 130
280 M$ = "MANEJO DE DISCO": HOME : INVERSE : HTAB (20 -  LEN (M$) / 2): PRINT M$: NORMAL
290 B$ = "
300  VTAB 6: CALL  - 958
310  HTAB 5: PRINT "1 GRABA GRAMATICA"
320  HTAB 5: PRINT "2 CARGA GRAMATICA"
330  HTAB 5: PRINT "3 SORRA GRAMATICA"
340  HTAB 5: PRINT "4 CATALOGO DE GRAMATICAS"
350  HTAB 5: PRINT "5 FIN"
360  VTAB 20: HTAB 1: CALL  - 958: PRINT "CUAL ";: GET R$: PRINT R$
370  FOR I = 1 TO 5: IF R$ =  STR$ (I) GOTO 390
380  NEXT I: GOTO 280
390  ON ( VAL (R$)) GOTO 410,670,900,1130,10
400  REM
410  VTAB 6: CALL  - 958: PRINT "GRABA GRAMATICA": IF B$(1) = "" THEN  PRINT "PRIMERO DEBES CAPTURAR GRAMATICA": FOR
     RR = 1 TO 400: NEXT RR: GOTO 280
420  VTAB 10: INPUT "NOMBRE ";M$: IF M$ = "" GOTO 410
430  IF  LEN (M$) > 20 THEN L = 20: GOSUB 1400: GOTO 410
440 C$ =  MID$ (M$,1,1)
450  IF  C$ < "A" OR  C$ > "Z" THEN  PRINT "NOMBRE INVALIDO": FOR RR = 1 TO 500: NEXT RR: GOTO 280
460 M$ = M$ +  MID$ (B$,1,20 -  LEN (M$))
470  FOR I = 1 TO T2: IF M$ = B$(I) THEN  PRINT "ESTA GRAMATICA YA EXISTE": FOR RR = 1 TO 500: NEXT I: GOTO 280
480  FOR I = 1 TO T6: IF  MID$ (G$(I),1,3) = "***" GOTO 500
490  NEXT I: PRINT "ARCHIVO PARA GRAMATICAS LLENO": FOR RR = 1 TO 500: NEXT : GOTO 280
500 B$(I) = M$:R$ =  STR$ (NT) +  MID$ (B$,1,2 -  LEN ( STR$ (NT))) +  STR$ (T) +  MID$ (B$,1,2 -  LEN ( STR$ (T))
    )
510  PRINT D$;"OPEN A-GRAMATICAS,L21,D2,C6"
520  PRINT D$;"WRITE A-GRAMATICAS,R";I: PRINT M$: PRINT D$;"WRITE A-GRAMATICAS,R0": PRINT R$: PRINT D$;"CLOSE A-GR
     AMATICAS"
530 M$ = "LETRA " +  STR$ (T)
```

```
540  FOR DR = 0 TO T + NT: PRINT D$;"OPEN"M$",L25,D2,S6": PRINT D$;"WRITE M$",R ;FR: PRINT E$(FR ; PRINT D$;"CLOSE
     "M$: NEXT : PRINT D$;"OPEN"M$",L25,D2,S6": PRINT D$; WRITE"M$",0: PRINT FR: PRINT D$;"CLOSE"M$
550  M$ = "ME" + STR$ (I)
560  FOR FR = 0 TO ME:FR$ = "": FOR R1 = 0 TO F1(FR,0) + 1:FR$ = FR$ + STR$ (F1(FR,R1)) + MID$ (B$,1,2 - LEN ( STR$
     (F1(FR,R1))): NEXT R1: PRINT D$;"OPEN"M$",L100,D2,S6": PRINT D$;"WRITE"M$",0": PRINT FR$: PRINT D$;"CLOSE"M
     $
570  NEXT
580  PRINT D$;"OPEN"M$",L100,D2,S6"
590  PRINT D$;"WRITE"M$",50": PRINT ME: PRINT D$;"CLOSE"M$
600  M$ = "MS" + STR$ (I)
610  FOR FR = 0 TO ME:FR$ = "": FOR R1 = 0 TO F1(FR,0) + 1:FR$ = FR$ + STR$ (F1(FR,R1)) + MID$ (B$,1,2 - LEN ( STR$
     (F1(FR,R1))): NEXT R1
620  PRINT D$;"OPEN "M$",L100,D2,S6": PRINT D$;"WRITE "M$",FR$: PRINT FR$: PRINT D$;"CLOSE"M$
630  NEXT FR
640  PRINT D$;"OPEN M$",L100,D2,S6"
650  PRINT D$; WRITE"M$",F0": PRINT MS: PRINT D$;"CLOSE"M$
660  GOTO 180
670  REM  ******CARGA GRAMATICA
680  VTAB 8: CALL  - 958: PRINT "CARGA GRAMATICA"
690  VTAB 10: INPUT "NOMBRE ";M$: IF M$ = "" GOTO 670
700  IF  LEN (M$) > 20 GOTO 670
710  IF  LEN (M$) < 20 THEN M$ = M$ +  MID$ (B$,1,20 -  LEN (M$))
720  FOR I = 1 TO T3: IF B$ (I) = M$ GOTO 740
730  NEXT I: PRINT "LA GRAMATICA ";: INVERSE : PRINT M$;: NORMAL : PRINT " NO EXISTE": FOR RR = 1 TO 400: NEXT RR:
      GOTO 280
740  M$ = "LETN" + STR$ (I)
750  PRINT D$;"OPEN"M$",L25,D2,S6": PRINT D$;"READ"M$",R0": INPUT R:NT =  VAL ( MID$ (E$,1,2)):T = VAL ( MID$ (R
     $,7,2))
760  FOR FR = 1 TO NT + T: PRINT D$;"READ"M$",R";RR: INPUT E$(FR): NEXT R: PRINT D$;"CLOSE"M$
770  M$ = "ME" + STR$ (I)
780  PRINT D$;"OPEN"M$",L100,D2,S6": PRINT D$;"READ"M$",F0": INPUT ME: PRINT D$;"CLOSE"M$
790  FOR RR = 1 TO ME: PRINT D$;"OPEN"M$",L100,D2,S6": PRINT D$;"READ"M$",R";RR: INPUT R$: PRINT D$;"CLOSE"M$
800  R =  - 1
810  FOR R1 = 1 TO  LEN (R$) STEP 2:R = R + 1:F(RR,R) =  VAL ( MID$ (R$,R1,2)): NEXT R1
820  NEXT RR
830  M$ = "MS" + STR$ (I)
840  PRINT D$;"OPEN"M$",L100,D2,S6": PRINT D$;"READ"M$",F0": INPUT MS: PRINT D$;"CLOSE"M$
850  FOR RR = 1 TO MS: PRINT D$;"OPEN"M$",L100,D2,S6": PRINT D$;"READ"M$",R";FR: INPUT R$: PRINT D$;"CLOSE"M$
860  VTAB 20: HTAB 1: PRINT "";S =  - 1: FOR R1 = 1 TO  LEN (R$) STEP 2:R = R + 1
870  P1(FR,R) =  VAL ( MID$ (R$,R1,2)): NEXT R1
880  NEXT RR
890  GOTO 220
900  REM ****** BORRA GRAMATICA
910  INVERSE :M$ = "BORRA GRAMATICA": HOME : HTAB (20 - ( LEN (M$) / 2)): PRINT M$: NORMAL
920  VTAB 10: INPUT "NOMBRE ";M$: IF M$ = "" GOTO 1)
930  IF  LEN (M$) > 20 THEN  PRINT "LONGITUD MAXIMA 20 CARACTERES": FOR RR = 1 TO 300: NEXT RR: GOTO 10
940  IF M$ = "" GOTO 10
950  M$ = M$ +  MID$ (B$,1,20 -  LEN (M$))
960  FOR I = 1 TO T6: IF B$(I) <  > M$ THEN  NEXT : PRINT "LA GRAMATICA ";: INVERSE : PRINT M$;: NORMAL : PRINT "
     NO EXISTE": FOR R = 1 TO 500: NEXT : GOTO 50
970  VTAB 10: HTAB 1: FLASH : PRINT "ESTAS SEGURO QUE QUIERES BORRAR ???";: PRINT : PRINT "LA GRAMATICA";: NORMAL
     : PRINT "  ";M$: PRINT "TECLEA S/N";: GET R$: PRINT R$: IF R$ = "" GOTO 970
980  IF R$ = "N" GOTO 280
990  IF R$ <  > "S" GOTO 97)
1000 B$(I) = "**********"
1010 PRINT D$;"OPEN A-GRAMATICAS,L26,D2,S6"
1020 PRINT D$;"WRITE A-GRAMATICAS,"I": PRINT "**********": PRINT D$;"CLOSE A-GRAMATICAS"
1030 RESTORE
1040 DATA "LETN","ME","MS"
1050 FOR J = 1 TO 3: READ M$:M$ = M$ +  STR$ (I)
1060 PRINT D$;"DELETE M$",D2,S6"
1070 NEXT J
1080 VTAB 2): CALL  - 958: VTAB 22: HTAB 1: PRINT "CONTINUAS S/N ";: GET R$: PRINT R$
1090 IF R$ = "S" GOTO 10
1100 IF R$ = "N" GOTO 1120
1110 GOTO 1080
```

```
1120 GOTO 230: REM  ****FIN DE BORRA
1130 REM  CATALOGO DE GRAMATICAS******
1140 M$="  CATALOGO DE GRAMATICAS"
1150 VTAB 1: INVERSE : HTAB (20 - ( LEN (M$) / 2)): PRINT M$: NORMAL
1160 VTAB 3: PRINT "POR:"
1170 VTAB 5: HTAB 7: PRINT "S PANTALLA"
1180 VTAB 7: HTAB 7: PRINT "I IMPRESOR"
1190 VTAB 9: HTAB 7: PRINT "F FIN"
1200 VTAB 20: HTAB 1: CALL  - 958: PRINT "CUAL";: GET R$: PRINT R$
1210 IF R$ = "" GOTO 1310
1220 IF P$ <  > "I" GOTO 1240
1230 PRINT D$;"PR#1"
1240 HOME : VTAB 1: INVERSE : HTAB (20) - ( LEN (M$) / 2)): PRINT M$: NORMAL
1250 FOR I = 1 TO TG
1260  IF  MID$ (G$(I),1,1) = "#" THEN  VTAB 3 + I: PRINT I;: HTAB 6: INVERSE : PRINT "( VACIO )": NORMAL : GOTO 1
     280
1270  VTAB 3 + I: PRINT I;: HTAB 6: INVERSE : PRINT G$(I): NORMAL
1280  NEXT I
1290  IF R$ = "I" THEN  PRINT D$;"PR#0": GOTO 1130
1300  VTAB 20: HTAB 30: FLASH : PRINT "RETURN": VTAB 21: HTAB 30: PRINT "PARA  ": VTAB 22: HTAB 30: NORMAL : PRINT
     "CONTINUAR";: GET R$: PRINT R$: GOTO 1130
1310  GOTO 280
1320  REM *****
1330  HOME
1340  REM  CAPTURA GRAMATICA
1350 M$ = "CAPTURA DE GRAMATICA"
1360  VTAB 1: INVERSE : HTAB (20 -  LEN (M$) / 2): PRINT M$: NORMAL
1370  REM
1380  VTAB 4: CALL  - 958: INPUT "NUMERO DE TERMINALES ";T: IF T = 0 GOTO 10
1390  IF T < 1 OR T > 50 GOTO 1380
1400  VTAB 6: CALL  - 958: INPUT "NUMERO DE NO TERMINALES ";NT: IF NT = 0 GOTO 10
1410  IF NT < 1 OR NT > 50 GOTO 1400
1420  IF NT + T > 101 GOTO 1380
1430  FOR I = 1 TO T
1440  VTAB 10: CALL  - 958: PRINT "DAME EL  TERMINAL  ";I;" ";: INPUT E$(I:
1450  GOSUB 1710: IF E = 1 GOTO 1440
1460  NEXT I
1470  FOR I = T + 1 TO NT + T
1480  VTAB 10: CALL  - 958: PRINT "DAME EL NO TERMINAL ";I - T;" ";: INPUT E$(I)
1490  GOSUB 1710: IF E = 1 GOTO 1480
1500  NEXT I
1510  VTAB 4: HTAB 1: CALL  - 958
1520  INPUT "NUMERO DE PRODUCCIONES ";NP
1530  IF NP < 1 OR NP > 20 GOTO 1510
1540 ME = NP:MS = NP: FOR I = 1 TO NP:P = 0
1550  VTAB 10: HTAB 1: CALL  - 958
1560  PRINT "PRODUCCION ";I;" ";: INPUT P$:P = 0
1570  IF P$ = "" GOTO 1550
1580 P$ = P$ + " "
1590 K = 1
1600  FOR J = K TO  LEN (P$)
1610  IF  MID$ (P$,J,1) = " " OR J =  LEN (P$) THEN P = P + 1: GOTO 1630
1620  NEXT J: PRINT "ELEMENTO INVALIDO": SPEED= 10: FOR R = 1 TO 10: NEXT : SPEED= 255: GOTO 1550
1630 C$ =  MID$ (P$,K,J - K)
1640  FOR B = 1 TO NT + T: IF E$(B) <  > C$ THEN  NEXT B: PRINT "ELEMENTO INVALIDO": FOR RR = 1 TO 500: NEXT RR: GOTO
     1550
1650  IF K = 1 AND B <  = T THEN  PRINT "ELEMENTO INVALIDO": FOR RR = 1 TO 500: NEXT RR: GOTO 1550
1660 P(I,P) = B:K = J + 1:P(I,0) = P
1670  IF P <  LEN (P$) GOTO 1600
1680 P(I,0) = P(I,0) - 1
1690  NEXT I
1700  GOTO 1770
1710 E = 0: IF I = 1 GOTO 1750
1720  FOR J = 1 TO I - 1
1730  IF E$(J) = E$(I) THEN E = 1
1740  NEXT J
```

```
1750  IF E = 1 THEN  VTAB 22: HTAB 1: CALL  - 958: PRINT "** ELEMENTO REPETIDO **": FOR RR = 1 TO 500: NEXT RR
1760  RETURN
1770  REM
1780  FOR I = 0 TO MS: FOR J = 0 TO P1(I,0) + 1:P1(I,J) = P1(I,J): NEXT J: NEXT I: GOTO 10
1790  REM                                        ELIMINA ELEMENTOS INUTILES

1800 L1 = 0
1810  REM  FORMA L1(...)
1820  FOR I = 1 TO MS
1830  FOR J = 2 TO P1(I,0) + 1
1840  IF P1(I,J) > T GOTO 1860
1850  NEXT J: GOSUB 2020
1860  NEXT I
1870  FOR I = 1 TO MS:B = 0
1880  FOR J = 2 TO P1(I,0) + 1
1890  IF P1(I,J) <  = T GOTO 1920
1900 B = 1: FOR X = 1 TO L1: IF L1(X) = P1(I,J) GOTO 1920
1910  NEXT X: GOTO 1930
1920  NEXT J: IF B = 1 THEN  GOSUB 2020
1930  NEXT I
1940  FOR I = 1 TO MS
1950  FOR J = 1 TO P1(I,0) + 1
1960  IF P1(I,J) <  = T GOTO 1990
1970  FOR X = 1 TO L1: IF L1(X) = P1(I,J) GOTO 1990
1980  NEXT X:P1(I,0) = 0
1990  NEXT J
2000  NEXT I
2010  GOTO 2070
2020  REM  *****INSERTA NOTERMINAL
2030  IF L1 = 0 THEN L1 = 1:L1(1) = P1(I,1): GOTO 2060
2040  FOR X = 1 TO L1: IF L1(X) = P1(I,1) GOTO 2060
2050  NEXT X:L1 = L1 + 1:L1(L1) = P1(I,1)
2060  RETURN
2070 L1 = 1
2080 L1(L1) = P1(I,1)
2090  FOR I = 1 TO MS
2100  IF P1(I,0) = 0 GOTO 2170
2110  FOR X = 1 TO L1: IF L1(X  <  ) P1(I,1) GOTO 2160
2120  FOR J = 2 TO P1(I,0) + 1
2130  FOR Y = 1 TO L1: IF L1(Y) = P1(I,J) GOTO 2150
2140  NEXT Y:L1 = L1 + 1:L1(L1) = P1(I,J)
2150  NEXT J
2160  NEXT X
2170  NEXT I
2180  FOR I = 1 TO MS
2190  FOR J = 1 TO P1(I,0) + 1: IF P1(I,J) <  = T GOTO 2220
2200  FOR X = 1 TO L1: IF P1(I,J) = L1(X) GOTO 2220
2210  NEXT X:P1(I,0) = 0: GOTO 2230
2220  NEXT J
2230  NEXT I
2240 L = 0
2250  FOR I = 1 TO MS
2260  IF P1(I,0) = 0 GOTO 2280
2270 L = L + 1: FOR J = 0 TO P1(I,0) + 1:P1(L,J) = P1(I,J): NEXT J
2280  NEXT I:MS = L
2290  REM
2300  GOTO 10
2310  REM  ELIMINA PRODUCCIONES -E
2320 L1 = 0
2330  FOR I = 1 TO MS
2340  IF P1(I,0) > 1 GOTO 2380
2350  IF P1(I,2) > (T) OR P1(I,2) <  ) (T) THEN  GOTO 2380
2360  IF P1(I,2) = (T) AND E$(P1(I,2)) <  ) "EPSILON" THEN  GOTO 2380
2370  GOSUB 2020: REM  INSERTA ELEMENTO IZQUIERDO EN LA LISTA
2380  NEXT I
2390  IF L1 = 0 THEN  GOTO 3240
```

```
2400  FOR I = 1 TO MS
2410  FOR J = 2 TO P1(I,0) + 1
2420  IF P1(I,J) <  = 7 GOTO 2460
2430  FOR X = 1 TO L1: IF L1(X) = P1(I,J) GOTO 2450
2440  NEXT X: GOTO 2460
2450  NEXT J: GOSUB 2020
2460  NEXT I
2470  FOR I = 1 TO MS
2480  IF P1(I,2) = (T) THEN P1(I,0) = 0
2490  NEXT I
2500  REM  GENERA COMBINACIONES PARA SELECCIONAR NUEVAS PRODUCCIONES
2510  REM  *** PASO 4)
2520  REM  ****************************************FORMA NUEVAS PRODUCCIONES
2530 PN = MS:TN = MS
2540  FOR I = 1 TO TN:E = 0
2550  IF P1(I,0) <  = 1 GOTO 2710
2560  FOR J = 2 TO P1(I,0) + 1: FOR K = 1 TO L1: IF P1(I,J) = L1(K) THEN E = E + 1:E(E) = J
2570  NEXT K
2580  NEXT J
2590 TT = PN
2600  FOR C1 = 1 TO (2 ^ E) - 1:PN = PN + 1
2610  FOR B = 0 TO P1(I,0) + 1:P1(PN,B) = P1(I,B.: NEXT B: NEXT C1:PN = TT
2620  FOR C = 0 TO E - 1
2630  FOR A = 1 TO E:PN = PN + 1
2640  FOR D = A TO A + C
2650  IF D > E THEN :P1(PN,E(I)) = 0
2660 P1(PN,E(D)) = 0
2670  NEXT D
2680  NEXT A
2690  NEXT C
2700 PN = PN - 1
2710  NEXT I
2720  REM                                                                    COMPRIME VERTICAL
      Y HORIZONTALMENTE
2730 L = MS:TT = MS
2740  FOR I = MS + 1 TO PN
2750  IF P1(I,0) = 0 GOTO 2810
2760  FOR J = 2 TO P1(I,0) + 1
2770  IF P1(I,J) = 0 THEN  FOR K = J TO P1(I,0):P1(I,K) = P1(I,K + 1): NEXT K:P1(I,0) = P1(I,0) - 1: GOTO 2750
2780  NEXT J
2790 L = L + 1
2800  FOR J = 0 TO P1(I,0) + 1:P1(L,J) = P1(I,J): NEXT J
2810  NEXT I
2820 MS = L
2830  REM

2840  REM                                                                    ELIMINA PRODUCCION
     ES REPETIDAS
2850  FOR I = 1 TO MS - 1
2860  IF P1(I,0) = 0 GOTO 2940
2870  FOR J = I + 1 TO MS
2880  IF P1(I,0) <  > P1(J,0) GOTO 2930
2890  FOR B = 1 TO P1(I,0) + 1
2900  IF P1(I,B) <  > P1(J,B) GOTO 2930
2910  NEXT B
2920 P1(J,0) = 0
2930  NEXT J
2940  NEXT I
2950 L = 0: FOR I = 1 TO MS
2960  IF P1(I,0) = 0 GOTO 2980
2970 L = L + 1: FOR J = 0 TO P1(I,0) + 1:P1(L,J) = P1(I,J): NEXT J
2980  NEXT I:MS = L
2990  REM *** PASO 5)
3000 P$ = "P":A = NT + 1 + 1
3010  FOR X = 1 TO L1
3020  IF P1(1,1) = L1(X) THEN PN = MS:E$(A) = E$(P1(1,1)) + P$:PN = PN + 1:P1(PN,1) = A:P1(PN,0) = P1(1,1):P1(PN,0
```

```
    , = 1:FN = FN + 1:P1(FN,1) = A:P1(FN,2) = 1:P1(FN,3) = 1
3470 NEXT I
3480 REM *** RECREA LA MATRIZ P1
3485 W = 1
3490 FOR I = MS + 1 TO FN
3500 W = W + 1
3505 FOR J = 0 TO P1(I,0) + 1
3090 PA(W,J) = P1(I,J)
3100 NEXT J
3110 NEXT I
3120 W = 0
3130 FOR I = 1 TO MS
3140 W = W + 1
3150 FOR J = 0 TO P1(I,0) + 1
3160 PA(W,J) = P1(I,J)
3170 NEXT J
3180 NEXT I
3190 FOR I = 1 TO FN
3200 FOR J = 0 TO PA(I,0) + 1
3210 P1(I,J) = PA(I,J)
3220 NEXT J
3230 NEXT I:MS = FN
3240 GOTO 50
3250 REM  **ELIMINA PRODUCCIONES SENCILLAS
3260 REM  PASO 1) 2  3)*****
3270 FN = MS:E = T + 1
3280 REM  PARA TODOS LOS N-T
3290 K = 0:L1 = 1
3300 L1(1) = E
3310 REM  REPITE 2 VECES*****
3320 FOR E = 1 TO MS
3330 IF P1(I,0) > 1 GOTO 3430
3340 IF P1(I,2) <  = T GOTO 3430
3350 S = 0: FOR X = 1 TO L1
3360 IF L1(X) = P1(I,1) THEN S = 1
3370 NEXT X
3380 IF S <  > 1 GOTO 3430
3390 S = 0: FOR X = 1 TO L1
3400 IF L1( X) = P1(I,2) THEN S = 1
3410 NEXT X
3420 IF S = 0 THEN L1 = L1 + 1:L1(L1) = P1(I,2)
3430 NEXT I
3440 K = K + 1: IF K < 2 GOTO 3310
3450 REM  ALMENTA NUEVA PRODUCIONES**********
3460 FOR I = 1 TO MS
3470 IF L1 < 2 GOTO 3570
3480 IF P1(I,0) = 1 AND P1(I,2) > T GOTO 3560
3490 FOR X = 2 TO L1
3500 IF P1(I,1) = L1(X) GOTO 3520
3510 NEXT X: GOTO 3570
3520 FN = FN + 1:P1(FN,1) = E
3530 FOR J = 2 TO P1(I,0) + 1
3540 P1(FN,J-1) = P1(I,J):P1(FN,0) = P1(I,0) - 1
3550 NEXT J
3560 NEXT I
3570 E = E + 1: IF (E <  = NT + T) OR (E <  = NT + T + 1) THEN  GOTO 3290
3590 MS = FN
3600 FOR I = 1 TO MS
3610 IF P1(I,0) = 1 AND P1(I,2  ) T THEN P1(I,0) = 0
3620 NEXT I
3630 L = 0
3640 FOR I = 1 TO MS
3645 IF P1(I,0) = 0 GOTO 3660
3650 L = L + 1: FOR J = 0 TO P1(I,0) + 1:P1(L,J) = P1(I,J): NEXT J
3660 NEXT I:MS = L
3670 GOTO 50
```

```
3690 REM *** FACTORIZA ***
3700 PN = MS:A = NT + 1:P$ = "P"
3710 FOR I = 1 TO MS - 1
3720 FOR J = I + 1 TO MS
3730 IF P1(I,1) <> P1(J,1) OR P1(I,2) <> P1(J,2) THEN GOTO 4050
3740 A = A + 1
3750 E$(A) = E$:P1(I,1)) + P$
3760 REM *** AUMENTA NUEVAS PRODUCCIONES
3770 PN = PN + 1
3780 P1(PN,1) = P1(I,1)
3790 Y = 0
3800 FOR K = 2 TO P1(I,0) + 1
3810 IF P1(I,K) = P1(J,K) THEN P1(PN,K) = P1(I,K):Y = Y + 1
3820 NEXT K
3825 S = 0: IF Y = P1(I,0) THEN P1(PN,P1(I,0) + 2) = A:P1(PN,0) = Y + 1: GOTO 3970
3830 P1(PN,P1(I,0) + 1) = A
3840 P1(PN,0) = Y + 1
3850 REM --------------------
3870 S = 1
3880 PN = PN + 1
3890 P1(PN,1) = A
3900 W = 1:Y = Y + 1
3910 FOR X = Y + 1 TO P1(I,0) + 1
3920 W = W + 1
3930 P1(PN,W) = P1(I,X)
3940 NEXT X
3950 P1(PN,0) = W - 1
3960 REM --------------------
3970 IF S = 1 THEN GOTO 4070
3980 PN = PN + 1
3990 P1(PN,1) = A
4000 W = 1:Y = Y + 1
4010 FOR X = Y + 1 TO P1(J,0) + 1
4020 W = W + 1
4030 P1(PN,W) = P1(J,X)
4040 NEXT X
4050 P1(PN,0) = W - 1
4060 REM --------------------
4070 PN = PN + 1:P1(PN,1) = A:P1(PN,2) = T:P1(PN,0) = 1
4080 NEXT J
4090 NEXT I
4100 REM ***ELIMINA PRODS. CON EL MISMO LADO IZQ. Y PREFIJOS COMUNES
4110 FOR I = 1 TO MS - 1
4120 S = 0: FOR J = I + 1 TO MS
4130 IF P1(I,1) = P1(J,1) AND P1(I,2) = P1(J,2) THEN S = 1:P1(J,0) = 0
4140 NEXT J
4150 IF S = 1 THEN P1(I,0) = 0
4160 NEXT I:MS = PN
4170 L = 0
4180 FOR I = 1 TO MS
4190 IF P1(I,0) = 0 GOTO 4210
4200 L = L + 1: FOR J = 0 TO P1(I,0) + 1:P1(L,J) = P1(I,J): NEXT J
4210 NEXT I:MS = L
4220 GOTO 10
4230 REM ELIMINA RECURSION POR LA IZQUIERDA
4240 A = NT + T:P$ = "P":PN = MS
4250 REM *** PASO 1)
4260 B = T + 1
4270 REM *** PASO 2)
4280 S = 0: FOR I = 1 TO MS
4290 IF P1(I,0) = 0 GOTO 4320
4300 IF P1(I,1) <> B GOTO 4320
4310 IF P1(I,1) = B AND P1(I,2) = B THEN S = 1:P1(I,2) = 0
4320 NEXT I
4330 REM  SI S=1 SE FORMAN NUEVAS PRODUCCIONES
4340 IF B <> 1 GOTO 4460
```

```
4300  REM ACUMULA NUEVO N-T
4310  A = A + 1
4320  B$(A) = E$+B$ + F$
4330  FOR I = 1 TO MS
4340  IF P1(I,0) = 0 GOTO 4440
4350  IF P1(I,0) = 1 AND P1(I,2) = T GOTO 4440
4360  IF P1 I,1) <  > B GOTO 4440
4420  IF P1(I,1) = B AND P1(I,2) <  > 0 THEN PN = PN + 1:P1(PN,1) = B: FOR J = 2 TO P1(I,0) + 1:F1(PN,J) = P1(I,J)
      :P1(PN,0) = P1(I,0) + 1: NEXT J:P1(PN,J) = A: GOTO 4440
4430  IF P1(I,1) = B AND P1(I,2) = 0 THEN PN = PN + 1:P1(PN,1) = A: FOR J = 3 TO P1(I,0) + 1:F1(PN,J - 1) = P1(I,J
      ):P1(PN,0) = P1(I,0) - 1: NEXT J:PN = PN + 1:P1(PN,1) = A: FOR K = 3 TO P1(I,0) + 1:F1(PN,K - 1) = P1(I,K):P1
      (PN,0) = P1(I,0): NEXT K:P1(PN,P1(I,0) + 1) = A
4440  NEXT I
4450  GOSUB 4670:MS = PN
4460  REM *** PASO 3)
4470  IF B = NT - 1 THEN  GOTO 4740
4480  B = B + 1:C = T + 1
4490  REM *** PASO 4)
4500  L1 = 0:S = 0
4510  FOR I = 1 TO MS
4520  IF P1(I,0) = 0 GOTO 4550
4530  IF P1(I,1) <  > B GOTO 4550
4540  IF P1(I,1) = B AND P1(I,2) = C THEN S = 1:P1(I,2) = 0: FOR J = 3 TO P1(I,0) + 1:L1 = L1 + 1:L1(L1) = P1(I,J)
      : NEXT J
4550  NEXT I
4560  IF S <  > 1 GOTO 4640
4570  FOR I = 1 TO MS
4580  IF P1(I,0) = 0 GOTO 4620
4590  IF P1(I,1) = 1 AND P1(I,2) = T GOTO 4620
4600  IF P1(I,1) <  > C GOTO 4620
4610  IF P1(I,1) = C THEN PN = PN + 1:F1(PN,1) = B: FOR J = 2 TO P1(I,0) + 1:P1(PN,J) = P1(I,J): NEXT J: FOR K = 1
      TO L1:P1(PN,K + P1(I,0) + 1) = L1(K):P1(PN,0) = P1(I,0) + L1: NEXT K
4620  NEXT I
4630  GOSUB 4670:MS = PN
4640  REM ***PASO 5)
4650  IF C = B - 1 GOTO 4270
4660 C = C + 1: GOTO 4490
4670  REM SUBRUTINA ELIMINA PROD. DE B QUE COMIENZAN CON B O C
4680  FOR I = 1 TO MS
4690  IF P1(I,0) = 0 GOTO 4720
4700  IF P1(I,1) <  > B GOTO 4720
4710  IF P1(I,1) = B AND P1(I,2) = 0 THEN P1(I,0) = 0
4720  NEXT I
4730  RETURN
4740 L = 0
4750  FOR I = 1 TO MS
4760  IF P1(I,0) = 0 GOTO 4790
4770 L = L + 1: FOR J = 0 TO P1(I,0) + 1:F1(L,J) = P1(I,J): NEXT J
4780  NEXT I:MS = L
4790  GOTO 10
4800  REM  ******
4810  REM  IMPRESION DE GRAMATICA
4820  NOME$ = "IMPRESION DE GRAMATICA"
4830  VTAB 1: INVERSE : HTAB (20 -  LEN (NM$) / 2): PRINT NM$: NORMAL
4840  VTAB 3: PRINT "OPCION:"
4850  VTAB 5: PRINT "P-PANTALLA"
4860  VTAB 7: HTAB 7: PRINT "I IMPRESORA"
4870  VTAB 9: HTAB 7: PRINT "F FIN"
4880  VTAB 20: HTAB 1: PRINT "CUAL "; GET R$: PRINT R$
4890  IF R$ = "F" GOTO 10
4900  IF R$ = "I" OR R$ = "P" GOTO 4920
4910  GOTO 4830
4920  VTAB 2: HTAB 1: CALL  - 958: PRINT "E=---ENTRADA": PRINT "S=---SALIDA": PRINT "A=---AMBAS": PRINT "
      CUAL";: GET E$
4930  PRINT E$
4940  IF E$ = "A" OR E$ = "E" OR E$ = "S" GOTO 4960
```

```
4950  GOTO 4920
4960  IF E$     "P" THEN  PRINT D$;"PR#0"
4970  IF E$     E" AND E$     A" GOTO 5100
4980  VTAB 1: INVERSE : HTAB (20 -  LEN (M$) / 2): PRINT M$: NORMAL : PRINT : PRINT "GRAMATICA DE ENTRADA": PRINT

4990  FOR I = 1 TO NE: VTAB 2: CALL  - 958
5000  IF P(I,0) =  GOTO E
5010  PRINT "PRODUCCION ";I: PRINT
5020  FOR J = 1 TO P(I,0) + 1
5030  IF J <  > 1 THEN  HTAB 10
5040  PRINT E$(P(I,J))
5050  NEXT J: IF R$ = "I" GOTO 5070
5060  VTAB 23: HTAB 1: INPUT "OPRIME RETURN PARA CONTINUAR ";RR$
5070  NEXT I
5080  REM
5090  IF E$ = "E" GOTO 5200
5100  VTAB 1: INVERSE : HTAB (20 -  LEN (N$) / 2): PRINT N$: NORMAL : PRINT : PRINT "GRAMATICA DE SALIDA": PRINT :
      PRINT
5110  FOR I = 1 TO NS: VTAB 2: CALL  - 958
5120  IF P(I,0) = 0 GOTO 5190
5130  PRINT "PRODUCCION ";I: PRINT
5140  FOR J = 1 TO P(I,0) + 1
5150  IF J <  > 1 THEN  HTAB 10
5160  PRINT E$(P(I,J))
5170  NEXT J: IF R$ = "I" GOTO 5190
5180  VTAB 23: HTAB 1: INPUT "OPRIME RETURN PARA CONTINUAR ";RR$
5190  NEXT I
5200  REM
5210  REM
5220  IF R$ = "I" THEN  PRINT D$;"PR#0"
5230  GOTO 10
5240  END
5250  HOME
5260  VTAB 10: HTAB 16: FLASH : PRINT "CUIDADO": NORMAL
5270  VTAB 12: HTAB 9: PRINT "LA GRAMATICA CAPTURADA"
5280  VTAB 14: HTAB 11: PRINT "YA HA SIDO GRABADA"
5290  VTAB 22: HTAB 2: CALL  - 958: PRINT "TECLEA S/N":: GET R$: PRINT R$: IF R$ = "" GOTO 5290
5300  IF R$ = "N" THEN : HOME : VTAB 12: HTAB 9: FLASH : PRINT "PRIMERO DEBES GRABARLA": NORMAL : FOR RR = 1 TO 70
      0: NEXT RR: GOTO 280
5310  IF R$ = "S" THEN : HOME : VTAB 12: HTAB 17: FLASH : PRINT "ESPERA": NORMAL : PRINT D$;"BLOAD CHAIN,A$22),D1":
      CALL 520"BTOP\D1,S6"
```

```
10 HOME
20 M$ = "GENERADOR DE TABLAS DE CONTROL"
30 INVERSE : VTAB 2: HTAB (20 - LEN(M$) / 2): PRINT M$: NORMAL
40 VTAB 12: HTAB 17: FLASH : PRINT "ESPERA": NORMAL
50 REM GENERADOR DE TABLAS DE CONTROL
60 DIM MT(20,20),FIRST(10,10),P(10),A(15,15),B(15,15),C(15,15),D(15,15),SS(10,10),STACK(15),MC(15,15)
70 REM *** PASO 1)
80 FOR I=1 TO MS: FOR J=0 TO P1(I,J) + 1:MT(I,J) = P1(I,J): NEXT J: NEXT I
90 REM PASO 1.A)
100 FOR I=1 TO MS
110 FOR J=2 TO MT(I,0) + 1
120 IF MT(I,J) < T THEN MT(I,0) = 0: GOTO 140
130 NEXT J
140 NEXT I
150 REM *** PASO 1.B)
160 L1 = 0
170 FOR I = 1 TO MS
180 IF MT(I,0) = 0 GOTO 240
190 FOR J = 2 TO MT(I,0) + 1
200 IF MT(I,J) > T GOTO 240
210 NEXT J: IF L1 = 0 THEN L1 = 1: L1(1) = MT(I,1): GOTO 240
220 FOR X = 1 TO L1: IF MT(I,1) = L1(X) GOTO 240
230 NEXT X: L1 = L1 + 1: L1(L1) = MT(I,1)
240 NEXT I
250 FOR I =1 TO MS
260 IF MT(I,0) = 0 GOTO 340
270 FOR J = 2 TO MT(I,0) + 1
280 IF MT(I,J) < = T GOTO 340
290 FOR X = 1 TO L1: IF MT(I,J) = L1(X) GOTO 310
300 NEXT X: GOTO 340
310 NEXT J
320 FOR Y =1 TO L1: IF MT(I,1) = L1(Y) GOTO 340
330 NEXT Y: L1 = L1 +1:L1(L1) = MT(I,1)
340 NEXT I
350 REM *** PASO 2)
360 REM COMIENZA DIRECTAMENTE CON(EDW)=MAT.A
370 FOR I=1 TO NT + T: FOR J = 1 TO NT +T:A(I,J) = 0: NEXT J: NEXT I
380 FOR I = 1 TO MS
390 IF P1(I,0) = 1 AND P1(I,2) = T GOTO 480
400 X = P1(I,1)
410 FOR J = 2 TO P1(I,0) + 1
420 Y = P1(I,J)
430 A(X,Y) = 1
440 FOR X = 1 TO L1
450 IF P1(I,J) < > L1(X) GOTO 480
460 NEXT X
470 NEXT J
480 NEXT I
490 REM *** PASO 3)
500 REM COMIENZA CON(EDW)=MAT.A
510 FOR J = 1 TO NT + T
520 FOR I = 1 TO NT + T
530 IF A(I,J) < > 1 GOTO 570
540 FOR K = 1 TO NT + T
550 IF A(J,K) = 1 THEN A(I,K) = 1
560 NEXT K
570 NEXT I
580 A(J,J) = 1
590 NEXT J
600 REM *** PASO 4), PASO 5)
610 REM PASO_4): FIRST DE C/NO-TER.
```

```
620 REM PASO 5): FIRST DE C/LADO DERECHO DE C/P
630 FOR I = 1 TO MS: FOR J = 1 TO T - 1: FIRST(I,J) = 0: NEXT J: NEXT I
640 FOR I = 1 TO MS
650 FOR A = 1 TO T - 1: F(A) = 0: NEXT A
660 IF P1(I,0) AND P1(I,2) = T THEN FIRST(I,0) = 0: GOTO 830
670 FOR J = 2 TO P1(I,0) + 1
680 IF P1(I,J) > T GOTO 720
690 FOR Y = 1 TO T - 1
700 IF P1(I,J) = Y THEN F(Y) = 1
710 NEXT Y: GOTO 800
720 A = P1(I,J)
730 FOR A = 1 TO T - 1
740 IF A(A,A) = 1 THEN F(A) = 1
750 NEXT A
760 FOR A = 1 TO L1
770 IF P1(I,J) < > L1(A) THEN GOTO 800
780 NEXT A
790 NEXT J
800 FOR A = 1 TO T - 1
810 FIRST(I,A) = F(A)
820 NEXT A: FIRST(I,0) = T - 1
830 NEXT I
840 REM *** PASO 6)
850 REM ESTA DIRECTAMENTE SEGUIDO POR(IFDB)=MAT,B
860 FOR I = 1 TO NT + T: FOR J = 1 TO NT + T: B(I,J) = 0:NEXT J: NEXT I
870 X = 0:Y = 0
880 FOR I = 1 TO MS: S = 0
890 IF P1(I,0) = 1 GOTO 1020
900 FOR J = 2 TO P1(I,0) + 1
910 X = P1(I,J)
920 FOR K = J + 1 TO P1(I,0) + 1
930 IF S = 1 THEN GOTO 1020
940 Y = P1(I,K)
950 B(X,Y) = 1
960 IF K = P1(I,0) + 1 THEN GOTO 1010
970 FOR X = 1 TO L1
980 IF L1(X) < > P1(I,K) THEN S = 1: GOTO 1010
990 NEXT X
1000 NEXT K
1010 NEXT J
1020 NEXT I
1030 REM *** PASO 7)
1040 REM ES DIRECTAMENTE EL FINAL DE(IDEO)=MAT,C
1050 FOR I = 1 TO NT + T: FOR J = 1 TO NT + T:C(I,J) = 0: NEXT J: NEXT I
1060 FOR I = 1 TO MS
1070 IF P1(I,0) = 1 AND P1(I,2) = T THEN GOTO 1160
1080 X = P1(I,1)
1090 FOR J = P1(I,0) + 1 TO 2 STEP - 1
1100 Y = P1(I,J)
1110 C(X,Y) = 1
1120 FOR X =1 TO L1
1130 IF P1(I,J) < > L1(X) GOTO 1160
1140 NEXT X
1150 NEXT J
1160 NEXT I
1170 REM *** PASO 8)
1180 REM ES EL FINAL DE(IEO)=MAT,C
1190 FOR J = 1 TO NT + T
1200 FOR I = 1 TO NT + T
1210 IF C(I,J) < > 1 GOTO 1250
1220 FOR K = 1 TO NT + T
1230 IF C(J,K) = 1 THEN C(I,K) = 1
1240 NEXT K
1250 NEXT I
1260 C(I,J) = 1
1270 NEXT J
```

```
1280 REM *** PASO 9)
1290 REM ESTA SEGUIDO POR(IFB)= MAT.MT
1300 FOR I = 1 TO NT + T: FOR J = 1 NT + T:MT(I,J) = 0:NEXT J: NEXT I
1310 REM PRODUCTO DE RALACIONES C.B=D
1320 FOR J = 1 TO NT + T
1330 FOR I = 1 TO NT + T
1340 IF C(I,J) < > 1 GOTO 1380
1350 FOR K = 1 TO NT + T
1360 IF B(I,K) = 1 THEN D(J,K) = 1
1370 NEXT K
1380 NEXT I
1390 NEXT J
1400 REM PRODUCTO DE RELACIONES D.A=MT
1410 FOR J = 1 TO NT + T
1420 FOR I = 1 TO NT + T
1430 IF D(I,J) < > 1 GOTO 1470
1440 FOR K = 1 TO NT + T
1450 IF A(J,K) = 1 THEN MT(I,K) = 1
1460 NEXT K
1470 NEXT I
1480 NEXT J
1490 REM *** PASO 10)
1500 REM AMPLIAR LA RELACION (IFB)
1510 FOR I = 1 TO NT + T
1520 MT(I,T) = C(I,1)
1530 NEXT I
1540 REM *** PASO 11), PASO 12)
1550 REM CONSTRUYE CONJUNTO DE SELECCION(SS) PARA C/P
1560 FOR I = 1 TO NS: FOR J = 1 TO T: SS(I,J) = 0: NEXT J: NEXT I
1570 X = 0:Y = 0
1580 FOR I = 1 TO NS:S = 0
1590 FOR A = 1 TO T: F(A) = 0: NEXT A
1600 IF P1(I,0) = 1 AND P1(I,2) = T THEN Y = P1(I,1): FOR W = 1 TO T: SS(I,W) = MT(Y,W): NEXT W: SS(I,0) = T:
GOTO 1790
1610 FOR J = 2 TO P1(I,0) + 1
1620 IF P1(I,J) > T GOTO 1670
1630 IF S = 1 GOTO 1760
1640 FOR W = 1 TO T
1650 IF P1(I,J) = W THEN F(W) = 1
1660 NEXT W: GOTO 1760
1670 X = P1(I,J)
1680 FOR W = 1 TO T
1690 IF A(X,W) = 1 THEN F(W) = 1
1700 NEXT W
1710 FOR X = 1 TO L1
1720 IF P1(I,J) < > L1(X) GOTO 1760
1730 NEXT X: S = 1
1740 IF J = P1(I,0) + 1 THEN Y = P1(I,1): FOR W = 1 TO T: IF MT(Y,W) THEN F(W) = 1: NEXT W
1750 NETX J
1760 FOR W = 1 TO T
1770 SS(I,W) = F(W)
1780 NEXT W: SS(I,0) = T
1790 NEXT I
1800 REM DETERMINA SI LA GRAMATICA ES LL(1:
1810 FOR I =1 TO NS - 1
1820 FOR J = I +1 TO NS
1830 IF P1(I,1) < > P1(J,1) GOTO 1870
1840 FOR K = 1 TO T
1850 IF SS(I,K) = 1 AND SS(J,K) = 1 GOTO 1920
1860 NEXT K
1870 NEXT J
1880 NEXT I
1890 VTAB 10: HTAB 3: CALL -958: FLASH: PRINT "LA GRAMATICA ANALIZADA SI ES LL(1)": NORMAL
1900 VTAB 12: HTAB 3: PRINT "POR LO TANTO SE PROCEDE A OBTENER"
1910 VTAB 14: HTAB 10: PRINT " LA TABLA DE CONTROL": FOR RR = 1 TO 600: NEXT RR: GOTO 1950
1920 VTAB 10: HTAB 3: CALL - 958: FLASH : PRINT "LA GRAMATICA ANALIZADA NO ES LL(1): NORMAL
```

```
1930 VTAB 12: HTAB 3: PRINT "POR LO TANTO NO ES POSIBLE OBTENER"
1940 VTAB 14: HTAB 10: PRINT "LA TABLA DE CONTROL": FOR RR = 1 TO 600:NEXT RR
1941 HOME: VTAB 12: HTAB 13: FLASH : PRINT "REGRESA A MENU": NORMAL
1942 VTAB 14: HTAB 16: FLASH : PRINT "PRINCIPAL":NORMAL
1943 PRINT D$;"RUN CABEZA",D1,S6"
1950 REM *** PASO 12)
1960 REM GENERA TABLA DE CONTROL
1970 CALL - 958
1980 FOR I = 1 TO NT: FOR J =1 TO T: MC(I,J) = 0: NEXT J: NEXT I
1990 REM RECONOCE TIPO DE PRODUCCION
2000 FOR I = 1 TO MS
2010 TP = 0
2020 REM POP-RETIENE
2030 IF P1(I,0) = 1 AND P1(I,2) = T THEN TP = 1 : GOTO 2100
2040 REM POP-AVANZA
2050 IF P1(I,0) = 1 AND P1(I,2) < T THEN TP = 2: GOTO 2100
2060 REM REP(ALFA)-AVANZA
2070 IF P1(I,0) > 1 AND P1(I,2) < 1 THEN TP = 3: GOTO 2100
2080 REM REP(BETA)-RETIENE
2090 IF P1(I,0) > 1 AND P1(I,2) > I THEN TP = 4: GOTO 2100
2100 REM LLENA TABLA DE CONTROL
2110 X = P1(I,1)
2120 FOR J = 1 TO SS(I,0)
2130 IF SS(I,J) = 1 THEN MC(X,J) = TP
2140 NEXT J
2150 NEXT I
2160 REM SELECCIONA TERMINALES DEL LENG.STACK
2170 FOR K = 1 TO T - 1
2180 FOR I = 1 TO MS
2190 IF P1(I,0) = 1 AND P1(I,2) < = T GOTO 2230
2200 FOR J = 3 TO P1(I,0) + 1
2210 IF P1(I,J) = K THEN MC(K,K) = 2: GOTO 2240
2220 NEXT J
2230 NEXT I
2240 NEXT K
2250 REM FDC = FIN DE CADENA
2260 F$ = "FDC"
2270 MC(T,T) = 5
2271 T$ = "TABLA DE CONTROL"
2272 HOME
2273 INVERSE: VTAB 2: HTAB (20 - LEN(T$) / 2): PRINT T$: NORMAL
2274 VTAB 4
2275 FOR I = 1 TO NT + T: HTAB 12: FOR J = 1 TO T: PRINT MC(I,J);" ";:NEXT J: PRINT: PRINT: NEXT I
2278 VATB 23: HTAB 2: CALL -958: INPUT "OPRIME RETURN PARA CONTINUAR";R$
2279 IF R$ < > "" GOTO 2278
2280 CNA$ = " "
2281 ASTACK = T + 1: REM INDICA SIMB.INIC. EN EL TOPE DEL STACK
2282 ALE = 0: REM INICIA APUNTADOR DE LENG. DE ENTRADA
2283 STACK(0) = T: REM INDICA FONDO DEL STACK
2284 VTAB 4: CALL -958
2310 VTAB 6: HTAB 1: PRINT "DAME LA CADENA";: INPUT CNA$
2315 IF CNA$ = "FIN" THEN : HOME: VTAB 12: HTAB 13: FLASH: PRINT "REGRESA A MENU": NORMAL : VTAB 14: HTAB 16: FLASH: PRINT "PRINCIPA
L": NORMAL: PRINT D$:"RUN CABEZA,D1,S6"
2320 CNA$ =CNA$ + F$
2330 K = 1
2340 FOR W = K TO LEN (CNA$)
2350 IF MID$ (CNA$,W,1) = " " OR W = LEN (CNA$) GOTO 2370
2360 NEXT W
2370 CTER$ = MID$(CNA$,K,W - K)
2380 IF CTER$ = "FDC" THEN C = T: GOTO 2420
2390 FOR C = 1 TO T - 1
2400 IF ES(C) = CTER$ GOTO 2420
2410 NEXT C: VTAB 18: HTAB 1: CALL - 958: PRINT "NO PERTENECE AL LENG.ENTR.";" ";: FLASH: PRINT CTER$: NORMAL: FOR RR = 1 TO 600: NE
XT RR: GOTO 2280
2420 S1 = MC(ASTACK,C)
2430 ON S1 GOTO 2510,2540,2570,2670,2820
```

```
2435 VTAB 3: CALL -958
2440 VTAB 12: HTAB 5: FLASH: PRINT "***ERROR***":NORMAL
2450 VTAB 14: HTAB 2: FLASH: PRINT "CADENA NO ACEPTADA": NORMAL
2460 VTAB 16: HTAB 2: PRINT CNA$
2470 VTAB 22: HTAB 2: CALL -958: PRINT "QUIERES PROBAR OTRA CADENA"
2480 VTAB 23: HTAB 2: PRINT "TECLEA S/N";" ";:GET R$: PRINT R$
2490 IF R$ = "S" GOTO 2280
2500 IF R$ = "N" THEN: HOME: VTAB 12: HTAB 13: FLASH: PRINT "REGRESA A MENU": NORMAL: VTAB 14: HTAB 16: FLASH:PRINT "PRINCIPAL":NORM
AL:PRINT D$;"RUN CABEZA,D1,S6"
2510 REM POP-RETIENE
2520 ALE = ALE -1
2530 ASTACK = STACK(ALE): GOTO 2420
2540 REM POP-AVANZA
2550 ALE = ALE - 1
2560 ASTACK = STACK(ALE): GOTO 2310
2570 REM REP(ALFA)-AVANZA
2580 IF ALE > 0 THEN ALE = ALE - 1
2590 I = 1
2600 IF P1(I,0) = 1 AND P1(I,2) < = T GOTO 2620
2610 IF ASTACK = P1(I,1) AND C = P1(I,2) THEN : FOR J = P(I,0) + 1 TO 3 STEP -1:ALE = ALE + 1: STACK(ALE) = P1(I,J): NEXT J
2620 I = I + 1: IF I < = MS GOTO 2600
2630 ASTACK = STACK(ALE): GOTO 2310
2640 REM REP(LETA)-RETIENE
2700 IF ALE > 0 THEN ALE = ALE - 1
2710 S = 0: FOR I = 1 TO MS
2720 IF S = 1 THEN I = MS: GOTO 2790
2730 IF P1(I,0) = 1 AND P1(I,2) < = T GOTO 2790
2740 IF ASTACK < > P1(I,2) OR P1(I,2) < T GOTO 2790
2750 S = 1: FOR J = P1(I,0) + 1 TO 2 STEP - 1
2760 ALE = ALE + 1
2770 STACK(ALE) = P1(I,J)
2780 NEXT J
2790 NEXT I
2800 ASTACK = STACK(ALE): GOTO 2420
2810 K = W + 1: GOTO 2340
2820 REM ACEPTA LA CADENA
2830 VTAB 3: CALL -958
2840 VTAB 13: HTAB 2: FLASH: PRINT "CADENA ACEPTADA": NORMAL
2850 VTAB 15:HTAB 2: PRINT CNA$
2860 VTAB 21: HTAB 2: PRINT "QUIERES PROBAR OTRA CADENA"
2870 VTAB 23: HTAB 2: PRINT "TECLEA S/N";" ";: GET R$: PRINT R$
2880 IF R$= "S" GOTO 2260
2890 IF R$ = "N" THEN : HOME: VTAB 12: HTAB 13: PRINT "REGRESA A MENU": NORMAL:
VTAB 14: HTAB 16: FLASH: PRINT "PRINCIPAL":NORMAL:PRINT D$;"RUN CABEZA,D1,S6"
```

EJEMPLO, FUNCION APLICADA :

2) ELIMINA ELEMENTOS INUTILES

IMPRESION DE GRAMATICA

GRAMATICA DE ENTRADA

PRODUCCION 1
DEC
LISTA

PRODUCCION 2
LISTA
VAR
TIPO

PRODUCCION 3
LISTA
VAR
REAL
ENTERO

PRODUCCION 4
TIPO
LISTA
TIPO

PRODUCCION 5
ARREGLO
LISTA
+

PRODUCCION 6
ARREGLO
EPSILON

IMPRESIÓN DE GRAMATICA

GRAMATICA DE SALIDA

PRODUCCION 1
DEC
LISTA

PRODUCCION 2
LISTA
VAR
REAL
ENTERO

EJEMPLO, FUNCION APLICADA :

3) ELIMINA PRODUCCIONES EPSILON

IMPRESION DE GRAMATICA

GRAMATICA DE ENTRADA

PRODUCCION 1
DEC
        LISTA
        VAR
        TIPO

PRODUCCION 2
DEC
        EPSILON

PRODUCCION 3
LISTA
        TIPO
        ARREGLO

PRODUCCION 4
LISTA
        VAR

PRODUCCION 5
TIPO
        REAL
        LISTA
        DEC

PRODUCCION 6
TIPO
        EPSILON

PRODUCCION 7
ARREGLO
        VAR
        TIPO
        TIPO
        ENTERO
        LISTA

PRODUCCION 8
ARREGLO
        EPSILON

IMPRESION DE GRAMATICA

GRAMATICA DE SALIDA

PRODUCCION 1
DECP
        DEC

```
PRODUCCION 2
DECP
        EPSILON

PRODUCCION 3
DEC
        LISTA
        VAR
        TIPO

PRODUCION 4
LISTA
        TIPO
        ARREGLO

PRODUCCION 5
LISTA
        VAR

PRODUCCION 6
TIPO
        REAL
        LISTA
        DEC

PRODUCCION 7
ARREGLO
        VAR
        TIPO
        TIPO
        ENTERO
        LISTA

PRODUCCION 8
DEC
        VAR
        TIPO

PRODUCCION 9
DEC
        LISTA
        VAR

PRODUCCION 10
DEC
        VAR

PRODUCCION 11
LISTA
        ARREGLO

PRODUCCION 12
LISTA
        TIPO
```

```
PRODUCCION 13
TIPO
        REAL
        DEC

PRODUCCION 14
TIPO
        REAL
        LISTA

PRODUCCION 15
TIPO
        REAL

PRODUCCION 16
ARREGLO
        VAR
        TIPO
        ENTERO
        LISTA

PRODUCCION 17
ARREGLO
        VAR
        TIPO
        TIPO
        ENTERO

PRODUCCION 18
ARREGLO
        VAR
        ENTERO
        LISTA

PRODUCCION 19
ARREGLO
        VAR
        TIPO
        ENTERO

PRODUCCION 20
ARREGLO
        VAR
        ENTERO
```

EJEMPLO, FUNCION APLICADA :

4) ELIMINA PRODUCCIONES SENCILLAS

IMPRESION DE GRAMATICA

GRAMATICA DE ENTRADA

```
PRODUCCION 1
DEC
        ASIGNA
        TIPO
        ENTERO

PRODUCCION 2
DEC
        LISTA

PRODUCCION 3
DEC
        EPSILON

PRODUCCION 4
LISTA
        DEFINE
        ASIGNA
        LISTA

PRODUCCION 5
LISTA
        ARREGLO

PRODUCCION 6
TIPO
        REAL
        ARREGLO

PRODUCCION 7
TIPO
        LISTA

PRODUCCION 8
ARREGLO
        VAR

PRODUCCION 9
ARREGLO
        DEFINE

PRODUCCION 10
DEFINE
        ASIGNA
        TIPO

PRODUCCION 11
DEFINE
        ETIQUETA
```

IMPRESION DE GRAMATICA

GRAMATICA DE SALIDA

PRODUCCION 1
DEC
        ASIGNA
        TIPO
        ENTERO

PRODUCCION 2
DEC
        EPSILON

PRODUCCION 3
LISTA
        DEFINE
        ASIGNA
        LISTA

PRODUCCION 4
TIPO
        REAL
        ARREGLO

PRODUCCION 5
ARREGLO
        VAR

PRODUCCION 6
DEFINE
        ASIGNA
        TIPO

PRODUCCION 7
DEFINE
        ETIQUETA

PRODUCCION 8
DEC
        DEFINE
        ASIGNA
        LISTA

PRODUCCION 9
DEC
        VAR

PRODUCCION 10
DEC
        ASIGNA
        TIPO

```
PRODUCCION 11
DEC
        ETIQUETA

PRODUCCION 12
LISTA
        VAR

PRODUCCION 13
LISTA
        ASIGNA
        TIPO

PRODUCCION 14
LISTA
        ETIQUETA

PRODUCCION 15
TIPO
        DEFINE
        ASIGNA
        LISTA

PRODUCCION 16
TIPO
        VAR

PRODUCCION 17
TIPO
        ASIGNA
        TIPO

PRODUCCION 18
TIPO
        ETIQUETA

PRODUCCION 19
ARREGLO
        ASIGNA
        TIPO

PRODUCCION 20
ARREGLO
        ETIQUETA
```

EJEMPLO, FUNCION APLICADA :

5) ELIMINA RECURSION POR LA IZQ.

IMPRESION DE GRAMATICA

GRAMATICA DE ENTRADA

```
PRODUCCION 1
LISTA
        TIPO
        ARREGLO

PRODUCCION 2
LISTA
        VAR

PRODUCCION 3
LISTA
        EPSILON

PRODUCCION 4
TIPO
        ARREGLO
        LISTA

PRODUCCION 5
TIPO
        LISTA
        REAL

PRODUCCION 6
ARREGLO
        LISTA
        TIPO
        *
PRODUCCION 7
ARREGLO
        ENTERO
        ENTERO

PRODUCCION 8
ARREGLO
        VAR
```

IMPRESION DE GRAMATICA

GRAMATICA DE SALIDA

```
PRODUCCION 1
LISTA
        TIPO
        ARREGLO

PRODUCCION 2
LISTA
        VAR
```

```
PRODUCCION 3
LISTA
        EPSILON

PRODUCCION 4
TIPO
        ARREGLO
        LISTA

PRODUCCION 5
ARREGLO
        ENTERO
        ENTERO

PRODUCCION 6
ARREGLO
        VAR

PRODUCCION 7
TIPO
        VAR
        REAL

PRODUCCION 8
TIPO
        ARREGLO
        LISTA
        TIPOP

PRODUCCION 9
TIPOP
        ARREGLO
        REAL

PRODUCCION 10
TIPOP
        ARREGLO
        REAL
        TIPOP

PRODUCCION 11
TIPO
        VAR
        REAL
        TIPOP

PRODUCCION 12
ARREGLO
        VAR
        TIPO

PRODUCCION 13
ARREGLO
        VAR
        REAL
```

```
        ARREGLO
        TIPO

PRODUCCION 14
ARREGLO
        VAR
        REAL
        TIPOP
        ARREGLO
        TIPO

PRODUCCION 15
ARREGLO
        ENTERO
        ENTERO
        ARREGLOP

PRODUCCION 16
ARREGLO
        VAR
        ARREGLOP

PRODUCCION 17
ARREGLO
        VAR
        TIPO
        ARREGLOP

PRODUCCION 18
ARREGLOP
        LISTA
        ARREGLO
        TIPO

PRODUCCION 19
ARREGLOP
        LISTA
        ARREGLO
        TIPO
        ARREGLOP

PRODUCCION 20
ARREGLO
        VAR
        REAL
        ARREGLO
        TIPO
        ARREGLOP

PRODUCCION 21
ARREGLOP
        LISTA
        TIPOP
        ARREGLO
        TIPO
```

```
PRODUCCION 22
ARREGLOP
        LISTA
        TIPOP
        ARREGLO
        TIPO
        ARREGLOP

PRODUCCION 23
ARREGLO
        VAR
        REAL
        TIPOP
        ARREGLO
        TIPO
        ARREGLOP
```

EJEMPLO, FUNCION APLICADA :

6) FACTORIZA

IMPRESION DE GRAMATICA

GRAMATICA DE ENTRADA

PRODUCCION 1
A
Y
C
X

PRODUCCION 2
A
Y
B

PRODUCCION 3
B
Y
C
Z

PRODUCCION 4
B
W

PRODUCCION 5
C
X
C
Y

PRODUCCION 6
C
X
n

PRODUCCION 7
C
X
B
Z

IMPRESION DE GRAMATICA

GRAMATICA DE SALIDA

PRODUCCION 1
B
Y
C
Z

PRODUCCION 2
B
W

PRODUCCION 3
A
Y
AP

PRODUCCION 4
AP
C
X

PRODUCCION 5
AP
B

PRODUCCION 6
C
X
CP

PRODUCCION 7
CP
C
Y

PRODUCCION 8
CP
A

PRODUCCION 9
CP
B
Z

EJEMPLO, FUNCIONES APLICADAS (SECUENCIALMENTE) :

2) ELIMINA ELEMENTOS INUTILES

3) ELIMINA PRODUCCIONES EPSILON

4) ELIMINA PRODUCCIONES SENCILLAS

5) ELIMINA RECURSION POR LA IZQ.

IMPRESION DE GRAMATICA

GRAMATICA DE ENTRADA

PRODUCCION 1
DEC
LISTA

PRODUCCION 2
DEC
DEC
REAL
ENTERO
TIPO

PRODUCCION 3
DEC
VAR

PRODUCCION 4
LISTA
TIPO
VAR
REAL

PRODUCCION 5
LISTA
DEFINE
REAL

PRODUCCION 6
TIPO
TIPO
TIPO
VAR
LISTA

PRODUCCION 7
TIPO
DEFINE
VAR
REAL

PRODUCCION 8
TIPO
REAL

PRODUCCION 9
ARREGLO
LISTA
REAL
DEFINE

PRODUCCION 10
ARREGLO
        EPSILON

PRODUCCION 11
DEFINE
        VAR
        TIPO
        DEFINE

PRODUCCION 12
DEFINE
        DEFINE
        VAR
        LISTA

IMPRESION DE GRAMATICA

GRAMATICA DE SALIDA

PRODUCCION 1
DEC
        LISTA

PRODUCCION 2
DEC
        DEC
        REAL
        ENTERO
        TIPO

PRODUCCION 3
DEC
        VAR

PRODUCCION 4
LISTA
        TIPO
        VAR
        REAL

PRODUCCION 5
TIPO
        TIPO
        TIPO
        VAR
        LISTA

PRODUCCION 6
TIPO
        REAL

IMPRESION DE GRAMATICA

GRAMATICA DE SALIDA

PRODUCCION 1
DEC
LISTA

PRODUCCION 2
DEC
DEC
REAL
ENTERO
TIPO

PRODUCCION 3
DEC
VAR

PRODUCCION 4
LISTA
TIPO
VAR
REAL

PRODUCCION 5
TIPO
TIPO
TIPO
VAR
LISTA

PRODUCCION 6
TIPO
REAL

IMPRESION DE GRAMATICA

GRAMATICA DE SALIDA

PRODUCCION 1
DEC
DEC
REAL
ENTERO
TIPO

PRODUCCION 2
DEC
VAR

PRODUCCION 3
LISTA
TIPO

```
        VAR
        REAL

PRODUCCION 4
TIPO
        TIPO
        TIPO
        VAR
        LISTA

PRODUCCION 5
TIPO
        REAL

PRODUCCION 6
DEC
        TIPO
        VAR
        REAL
```

IMPRESION DE GRAMATICA

GRAMATICA DE SALIDA

```
PRODUCCION 1
DEC
        VAR

PRODUCCION 2
LISTA
        TIPO
        VAR
        REAL

PRODUCCION 3
TIPO
        REAL

PRODUCCION 4
DEC
        TIPO
        VAR
        REAL

PRODUCCION 5
DECP
        REAL
        ENTERO
        TIPO

PRODUCCION 6
DECP
        REAL
        ENTERO
```

```
        TIPO
        DECP

PRODUCCION 7
DEC
        VAR
        DECP

PRODUCCION 8
DEC
        TIPO
        VAR
        REAL
        DECP

PRODUCCION 9
TIPO
        TIPO
        VAR
        LISTA

PRODUCCION 10
TIPOP
        TIPO
        VAR
        LISTA
        TIPOP

PRODUCCION 11
TIPO
        REAL
        TIPOP
```

EJEMPLO, FUNCION APLICADA :

7) GENERA TABLA DE CONTROL

La tabla de control generada por el sistema a partir de la gramática de entrada, cuyo ejemplo se muestra a continuación, consiste de una tabla o matriz, donde el subindice (i) de cada renglon de la matriz, corresponde a la posición de los terminales y no-terminales almacenados en la lista. De estos elementos se hace una selección de terminales (que aparecen en otra posición que no sea el inicio de una producción), más los no-terminales; formando así el lenguaje del stack.
El lenguaje de entrada, se forma por todos los terminales de la gramática, cuya posición corresponde al subindice (j) de cada columna de la tabla. Una vez terminada la selección se procede a llenar la tabla de control, analizando cada producción de la gramática (asociada a su respectivo conjunto de selección). Ya completada la tabla, el sistema solicita la cadena a ser analizada. Como punto de inicio se toma el símbolo inicial (de la gramática) y primer terminal de la cadena, y de acuerdo al número de subrutina asignado a la posición direccionada en la tabla,se ejecuta unicamente la acción que corresponde como: Pop-Avanza,Pop-Retiene,Rep(alfa)-Avanza,Rep(beta)-Retiene. Para lo cual se auxilia de un stack sobre el cual se ejecuta la acción, este proceso se repite hasta determinar si la cadena es aceptada o no por el autómata.

GRAMATICA DE ENTRADA

PRODUCCION 1
DEC
VAR
LISTA
TIPO

PRODUCCION 2
DEC
EPSILON

PRODUCCION 3
LISTA
REAL
DEC
LISTA

PRODUCCION 4
LISTA
ENTERO
ENTERO
ENTERO

```
PRODUCCION 5
LISTA
                ASIGNA

PRODUCCION 6
TIPO
                LISTA
                VAR
                TIPO

PRODUCCION 6
TIPO
                VAR
```

UNA VEZ ANALIZADA LA GRAMATICA EL SISTEMA PRODUCE EL SIGUIENTE MENSAJE :

```
LA GRAMTICA ANALIZADA SI ES LL(1)
POR LO TANTO SE PROCEDE A OBTENER
     LA TABLA DE CONTROL
```

TABLA DE CONTROL

2	0	0	0	0
0	0	0	0	0
0	0	2	0	0
0	0	0	0	0
0	0	0	0	5
3	1	1	1	1
0	3	3	2	0
2	4	4	4	0

9. CONCLUSIONES

El sistema manipulador de gramaticas desarrollado en la microcomputadora APPLE II-E, de acuerdo a sus características, ofrece como resultado ser un sistema bastante confiable y flexible para ser implantando en cualquier institución que cuente con equipo de las mismas caracteristicas, ya que el grado de inteligencia que ofrece hacia el usuario le permite ser utilizado con gran facilidad.

A lo largo de este trabajo nos limitamos a las consideraciones de software y hardware para el diseno del sistema, por lo que queda abierta la posibilidad para la elaboración de otros trabajos futuros, en los cuales se disponga de más elementos de Software y Hardware, con el proposito de que el nuevo sistema pueda ser utilizado en equipos de computo con características de multiusuario, lo cual aporta un beneficio mucho mayor a cualquier institución donde pueda ser aplicado.

V BIBLIOGRAFIA

Compiler Design Theory. Philip M. Lewis,
Daniel J. Rosenkrantz, Richard E. Stearns, Addison-Wesley.

The Theory of Parsing, Translation and Compiling.
Vomunem 1 y II. Alfred V. Aho. Jeffrey D. Ullman.
Prentice-Hall, Inc.

Principles of Compiler Design. Alfred V. Aho.
Jeffrey D. Ullman. Addison-Wesley.

Programación de Sistemas. John J. Donovan.
Editorial "El Ateneo", Buenos Aires.

A Compiler Generator. William M. Mckuman.
James J. Horning. David B. Wortman.
Prentice-Hall, Inc.

Yet Another Compiler-Compiler: Yacc. Murray Hill.
A Lexical Analyzer Generator: Lex. Murray. Hill.