3. DISEÑO DEL INTERPRETE.

3.1. DESCRIPCION GENERAL DEL INTERPRETE QENQO.

En este capítulo se describen las características del intérprete.

3.1.1. TIPOS DE DATOS.

Se han establecido cuatro tipos de datos para definir variables:

a) NIVELES: Son variables que se integran en el tiempo.

b) FLUJOS : Son las variables que interactúan directamente con los niveles.

c) AUXILIARES: Sirven para operaciones intermedias.

d) TABLAS : Arreglos o listas de variables auxiliares con la finalidad de hacer interpolaciones.

El almacenamiento interno de los valores de todas las variables es en el tipo Real.

3.1.2. CONSTANTES Y PARAMETROS DE CONTROL.

3.1.2.1. CONSTANTES.

Se pueden definir constantes para ser utilizadas en cualquier parte del código.

3.1.2.2. PARAMETROS DE CONTROL.

Los parámetros de control son las variables internas que controlan la simulación del modelo; lo constituyen el horizonte de integración (desde un tiempo inicial, hasta un tiempo final), el intervalo dt para cada operación (dt >= 0), y las condiciones iniciales del modelo (valores iniciales de las variables de nivel y las tablas).

Además, se dispone de una variable global interna que controla el tiempo de simulación.

3.1.3. OPERACIONES.

Básicamente el intérprete puede soportar tres tipos de operaciones.

3.1.3.1. OPERACIONES CON VARIABLES DE NIVEL.

A una variable de nivel solamente se le pueden asignar expresiones que contengan variables de flujo, de nivel o constantes numéricas (expresiones de nivel):

variable de Nivel := expresión de Nivel.

Ejemplo: Nivel1 := Flujo1-Flujo2+Flujo3 - Nivel1;

3.1.3.2. OPERACIONES CON VARIABLES DE FLUJO Y AUXILIARES.

A una variable de flujo o auxiliar se le pueden asignar una expresión, conteniendo cualquier otra variable:

Variable de Flujo = Expresión cualquiera.

Ejemplo: Flujo1 = Var1*Nivel1-20;

3.1.3.3. OPERACIONES CON TABLAS.

Estas facilidades no están disponibles en esta versión del intérprete, pero pueden implementarse en el futuro. Más información puede encontrarse en [6].

3.1.4. SUBSISTEMAS.

Un sistema puede subdividirse, de acuerdo a su complejidad y al criterio del analista, en uno o más subsistemas [11], en cada uno de los cuales se pueden declarar constantes, variables y funciones propias. Esto se ha pensado así para mejorar el entendimiento de sistemas complejos. Cada subsistema puede entregar información a otro, mediante el valor de una o más variables, sin que estos valores puedan modificarse desde fuera del subsistema. A este modo de trabajo lo denomino 'modelación estructurada'.

3.1.5. FUNCIONES.

Existen predefinidas algunas funciones matemáticas comunes; asimismo el usuario puede definir sus propias funciones y utilizarlas en alguna parte de la descripción del modelo.

3.1.6. BIFURCACIONES.

El intérprete tiene la capacidad de realizar cambios en la el flujo de la descripción o cálculo de un modelo mediante la verificación de la validez de una expresión lógica.

Ejemplo:

Si <expresión lógica> **entonces** <Proposición 1>
Sino <Proposición 2>

3.1.7. ITERACIONES.

Repetir la ejecución de una o un grupo de instrucciones mientras sea válida una expresión lógica.

Ejemplo : **Mientras** <Expresión Lógica> **hacer**
<Proposición 1>

3.1.8. ORDENES PREDEFINIDAS.

Se construyen proposiciones predefinidas para ordenar la definición y la traza de gráficos con los datos generados o resultados.

Ejemplo :

Grafica(<Nombre de Variable>, <Tipo de Trazado>, <Límites>)

3.2. IMPLEMENTACION DEL LENGUAJE QENQO.

Se ha implementado utilizando el Lenguaje Turbo Pascal versión 4.0 para microcomputadoras IBM/PC y compatibles. Esto para lograr que una mayor cantidad de investigadores puedan usar el software.

3.2.1. GENERALIDADES.

El lenguaje Qenqo es un sistema de notación para describir un sistema mediante las relaciones entre sus variables y obtener, bajo ciertas condiciones iniciales, los valores de estas variables en el tiempo.

Un lenguaje es, en principio, una especificación

completa de los siguientes tres objetos:

a) Un conjunto de Palabras Reservadas (S),

b) Un conjunto de oraciones construídas de forma adecuada con estas palabras (S*), y

c) El significado de estas oraciones.

Las Palabras Reservadas del lenguaje Qenqo son las siguientes:

DE CONTROL	DE PROPOSICION	FUNCIONES
SISTEMA	NIVEL	ABS
CONST	FLUJO	ARCTAN
VAR	AUX	COS
CONTROL	TABLA	EXP
DESDE_T	INICIO	FRAC
HASTA_T	FIN	ENT
DT	MIENTRAS	CUAD
REPETICION	HACER	LN
	SI	LOG
	SINO	RAIZC
	ENTONCES	RND
		SEN

3.2.2. ESTRUCTURAS DEL LENGUAJE.

Para expresar un modelo en el lenguaje Q, se utilizan las siguientes estructuras:

- Definiciones y declaraciones.
- Expresiones.
- Bloques y ordenes.
- Definición e Invocación a Funciones y Subsistemas.

La descripción del sistema se empieza poniendo un nombre al sistema de la siguiente manera

Sistema <Nombre del Sistema>

3.2.2.1. DEFINICIONES Y DECLARACIONES.

Se declaran las constantes de la siguiente manera:

CONST <Identificador de Constante> = <Valor Real>

Las variables se declaran en siguiente forma:

VAR <Identificador de Variable> = <Tipo>

Donde el <Tipo> es de Dinámica de Sistemas.

3.2.2.2. Expresiones.

La expresiones son aquellas sentencias que son evaluadas. Contienen los siguientes componentes:

a) Operadores: + (Suma), - (Resta), * (Multiplicación), / (División).

b) Paréntesis: ().

c) Identificadores: Tira de caracteres diferentes a las Palabras Reservadas (S).

3.2.2.3. PROPOSICIONES.

Las proposiciones u ordenes hacen cambiar al estado de la máquina, en una o más operaciones.

En el lenguaje Qenqo, al igual que en Pascal existe composición secuencial de órdenes, diferenciadas entre ellas por el símbolo Punto Y Coma (;), como sigue:

Proposición_1;
Proposición_2;
...; Proposición_N;

donde, Proposición_1 se realiza primero, luego Proposición_2, así hasta Proposición_N.

Asimismo existen ordenes para la selección condicional y de iteración.

La proposición de mayór jerarquía con el símbolo INICIO, y termina cuando se encuentra el símbolo FIN, de la siguiente manera:

Inicio <Proposición> **Fin**

3.2.2.4. DEFINICION E INVOCACION A FUNCIONES Y SUBSISTEMAS.

En este lenguaje, la definición de los subsistemas y funciones de un sistema debe hacerse antes del sistema jerárquicamente que los llame. En el bloque principal o del subsistema mayor, la ejecución se realiza secuencialmente

La invocación a un subsistema o función se hace mediante la llamada a su nombre y de sus parámetros previamente

definidos.

3.2.2.5. ASIGNACIONES.

Las asignaciones de variables en Qenqo son de dos tipos que difieren en la sintaxis (en el símbolo para la asignación) y en la semántica (resultante de su operación).

Asignación a variables de Nivel:

<Variable de Nivel> := <Expresión de nivel>.

<Expresión de Nivel> equivale a:

$\langle \text{Variable de Nivel} \rangle_t = \langle \text{Variable de Nivel} \rangle_{t-1} + Dt * \langle \text{Expresión de Nivel} \rangle$

Asignación a Variables-No-Nivel:

<Variable no nivel> = <Expresión>.

a esta <Expresión> se hace una evaluación simple.

3.2.3. GRAMATICA.

Una gramática está definida por la cuádrupla (St,Sn,P,I):

a) Lista de **símbolos terminales (St).**

b) Lista de **símbolos no terminales (Sn)**, donde:

$Sn \cap St = \{\}$ o Nulo

$S = Sn \cup St$; S = Palabras Reservadas

c) Un conjunto de **reglas de Producción** formados por los pares (α,β), donde α está en S^+, y β está en S^*.

$\alpha \dashrightarrow \beta$.

d) **Un símbolo Inicial** que marca el inicio de la producción de cualquier sentencia o proposición del lenguaje.

3.2.4. El LENGUAJE.

Es el conjunto de todas las oraciones bien construídas con la gramática. Según [15], por la definición de un

lenguaje de programación se entiende a la descripción completa de la sintaxis y la semántica del lenguaje.

La sintaxis básica del Lenguaje Qenqo, se puede expresar en el Metalenguaje BNF Extendido (Backus-Naur Form) [13].

Símbolo Inicial: <Modelo>

Las siguientes son las reglas de producción del lenguaje.

```
<Modelo>          ::= Sistema <Ident.> <Bloque>.

<Bloque>          ::= [<Constantes>]
                      [<Variables>]
                      [<Controles>]
                      [<Funciones>]
                      [<Subsistemas>]
                      <Proposición>

<Constantes>      ::= const <Ident.> = <Número> ;
                      {<Ident> = <Número> ;}

<Variables>       ::= <Ident> {, <Ident> } : <Tipo>;
                      {<Ident> {, <Ident> } : <Tipo>;}

<Controles>       ::= control <Ident.Control> = <Número>;
                           {<Ident.Control> = <Número>;}
                           [<Ident.> = <Número>;
                             {<Ident> = <Número> ;}]

<Funciones>       ::= función <Ident> [<Lista parámetr>]:
                                 <Tipo> ;
                            <Bloque>;

<Subsistemas>     ::= subsistema <Ident>
                                 [<Lista parámetr>];
                              <Bloque>;


<Proposición>     ::= Inicio <proposición>;
                             {proposición ;}
                      Fin

<Lista parámetr>  ::= ( <Ident.> [{, <Ident> }] : <Tipo>
                      {; <Ident.>[{, <Ident> }]: <Tipo>} )

<Proposición>     ::= <Orden simple> |
                      <Orden estructurada>

<Orden simple>    ::= <Asignación> |
                      <Llamar subsistema> |
```

<Orden estructurada>::= <Proposición> ¦
<Orden Condicional> ¦
<Orden Repetitiva>

<Orden repetitiva> ::= **Mientras** <Expresión-Lógica>
Hacer
<Proposición>

<Orden condicional> ::= **Si** <Expresión Lógica> **Entonces**
<Proposición> [**Sino** <Proposición>]

<Asignación> ::= <Asig. Var. No Nivel> ¦
<Asig. Var. Nivel>

<Asig. Var No Nivel> ::= <Ident> = <Expresión>

<Asig. Var Nivel> ::= <Ident> := <Expresión>

<Llamar subsistema> ::= <Ident.> [<Lista expresión>]

<Lista expresión> ::= <Expresión> ¦ <Lista
expresión>, <Ident.>

<Expresión> ::= <Termino> ¦ <Expresión> + <Termino>
¦ <Expresión> - <Termino>

<Termino> ::= <Factor> ¦ <Termino> * <Factor>
¦ <Termino> / <Factor>

<Factor> ::= (<Expresión>) ¦ <Ident>
¦ <Número> ¦ <Llamar Función>

<Llamar Función> ::= <Ident.> [<Lista variables>]

<Expresión Lógica>::= <Termino Lógico>
¦<Expresión Lógica> **o** <Expresión Lógica>

<Termino Lógico> ::= <Factor Lógico>
¦<Termino Lógico> **y** <Termino Lógico>

<Factor Lógico> ::= (<Expresión Lógica>)
¦ <Ident> <Operador Lógico> <Ident>

<Operador Lógico>::= > ¦ >= ¦ < ¦ <= ¦ <> ¦ =

<Tipo> ::= Nivel ¦ Aux ¦ Flujo

<Ident.> ::= <Letra> { <Letra> ¦ <Dígito> }

<Letra> ::= A¦B¦C¦D¦E¦F¦G¦H¦I¦J¦K¦L¦M
¦N¦O¦P¦Q¦R¦S¦T¦U¦V¦W¦X¦Y¦Z

<Número> ::= [+ | -] <Entero sin signo>
[<Punto> <Entero sin signo>]

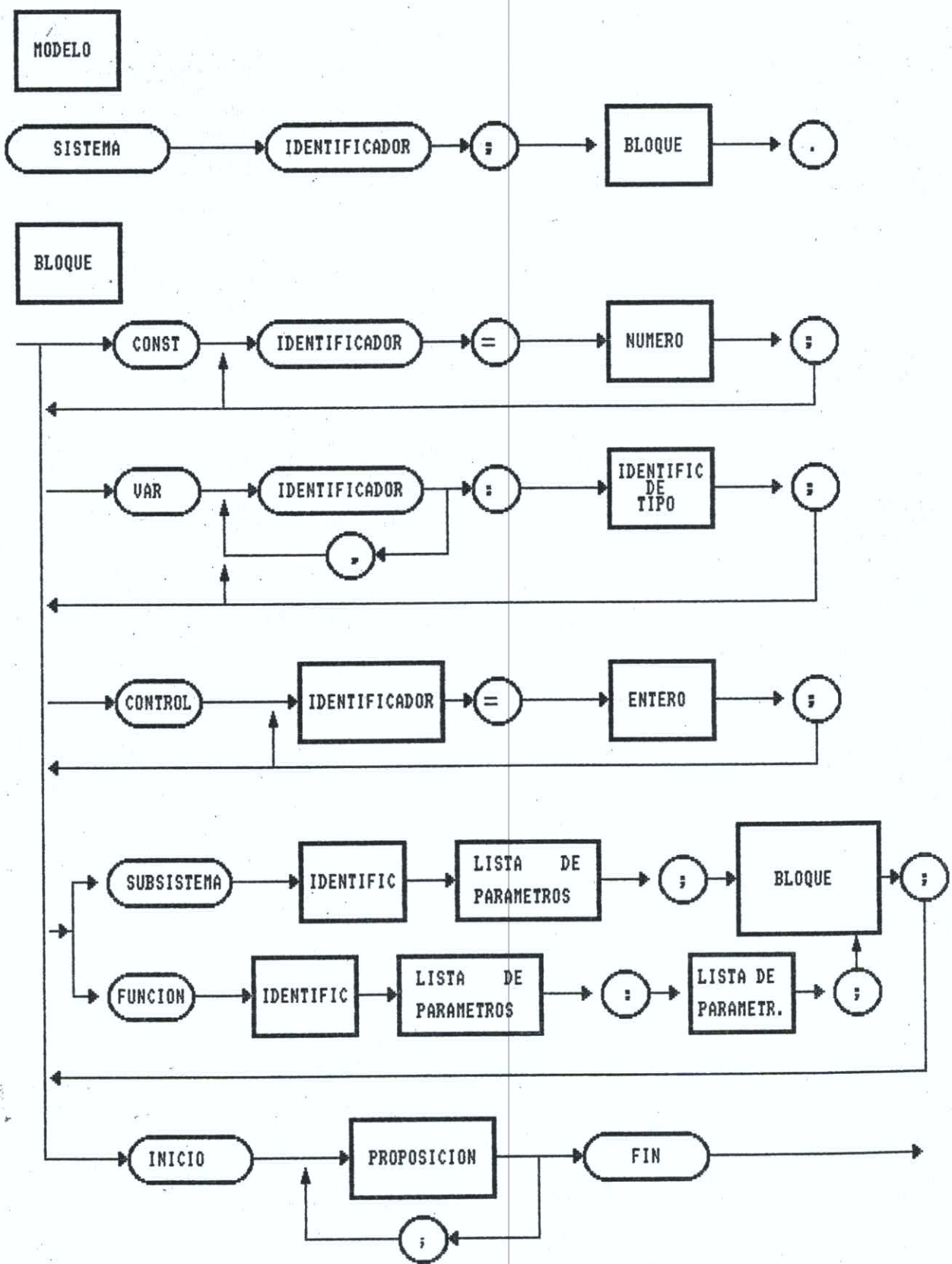
<Entero sin signo>::= <Dígito> { <Dígito> }

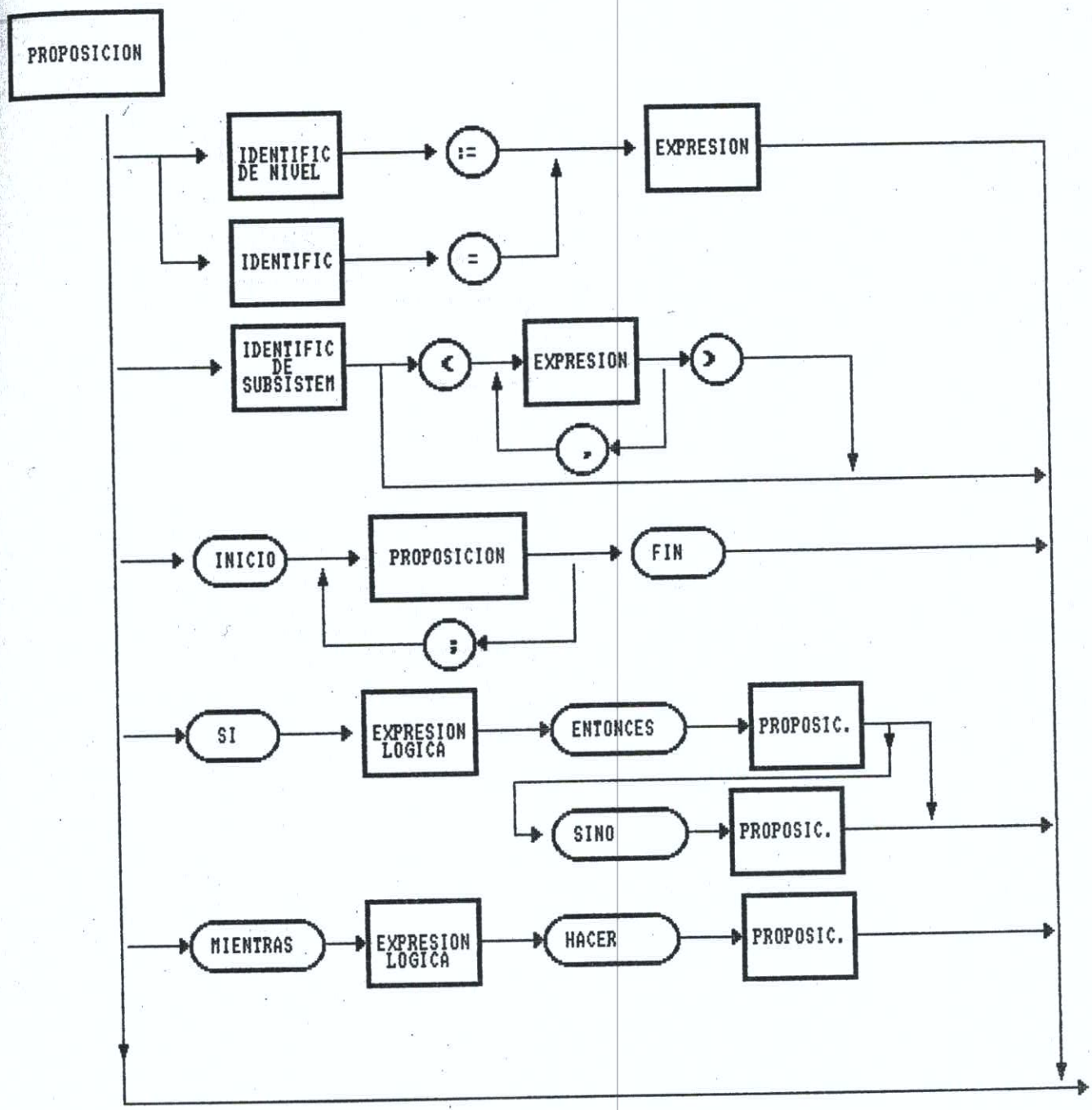
<Dígito> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9|0

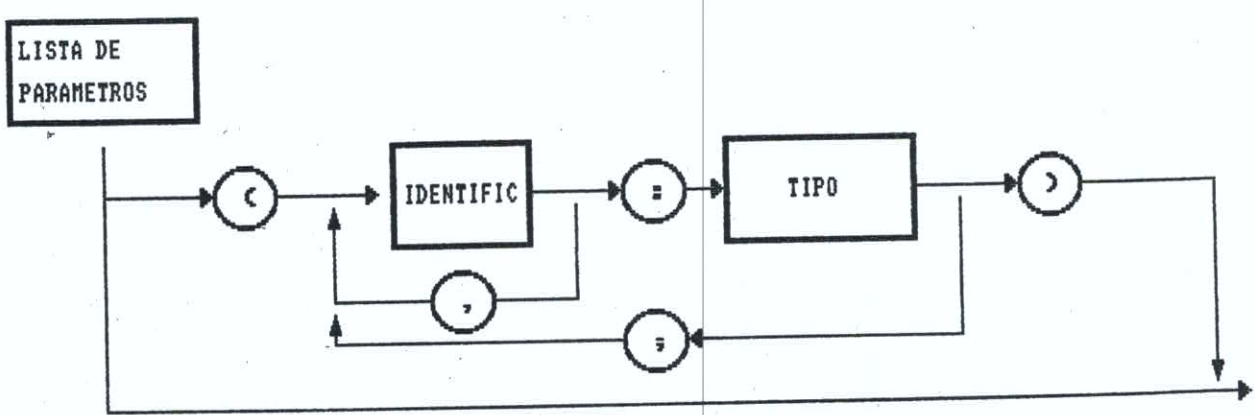
<Punto> ::= .

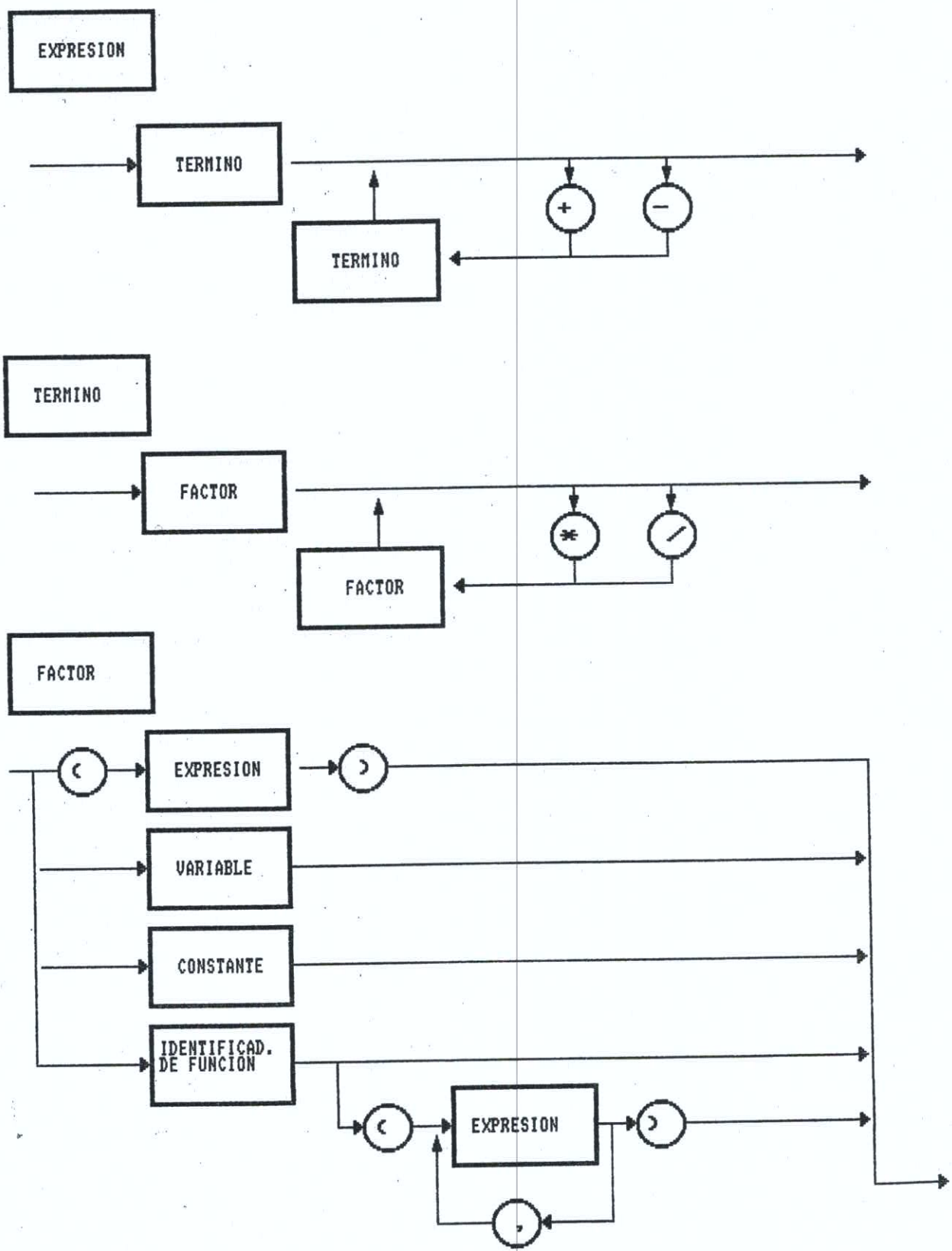
3.2.5. DIAGRAMAS DE CONWAY.

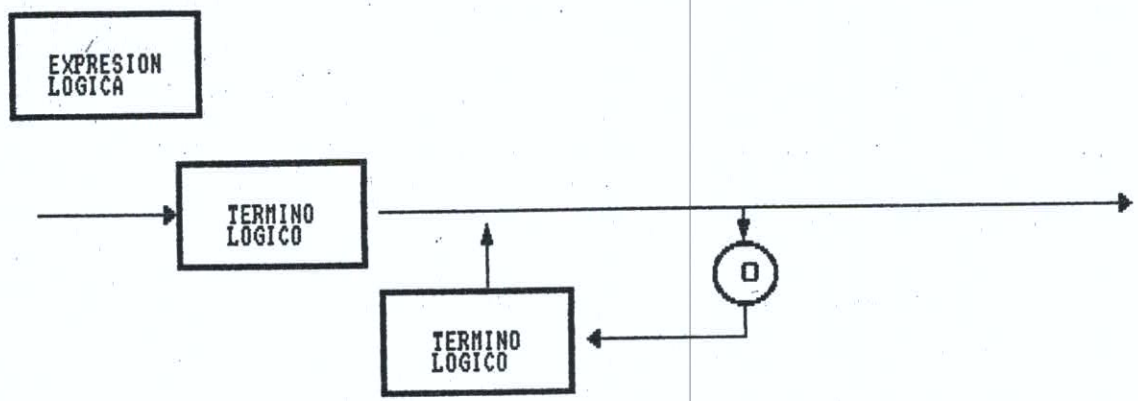
DIAGRAMA SINTACTICO DEL LENGUAJE QENQO

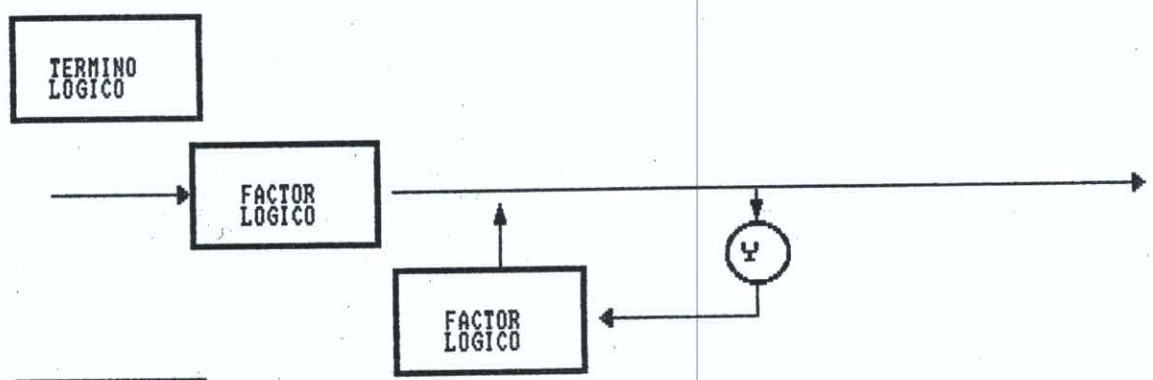


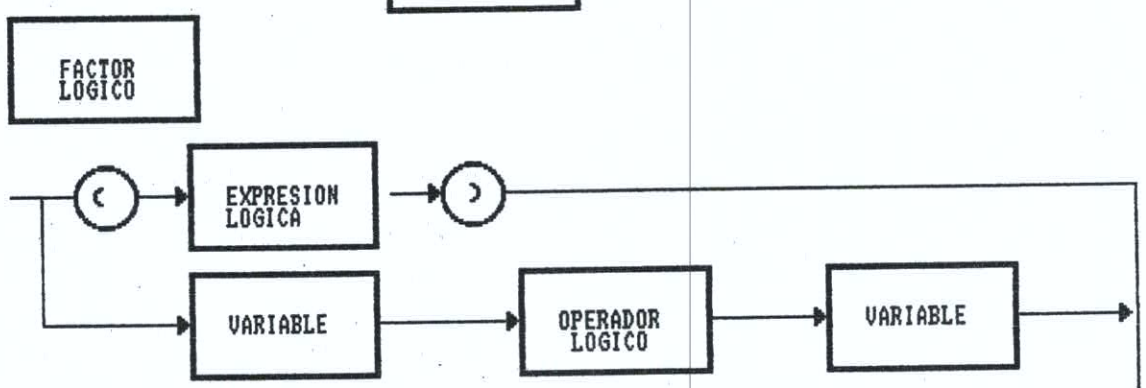

PROPOSICION
IDENTIFIC DE NIVEL
:=
EXPRESION
IDENTIFIC
=
IDENTIFIC DE SUBSISTEM
<
EXPRESION
>
,
INICIO
PROPOSICION
FIN
;
SI
EXPRESION LOGICA
ENTONCES
PROPOSIC.
SINO
PROPOSIC.
MIENTRAS
EXPRESION LOGICA
HACER
PROPOSIC.

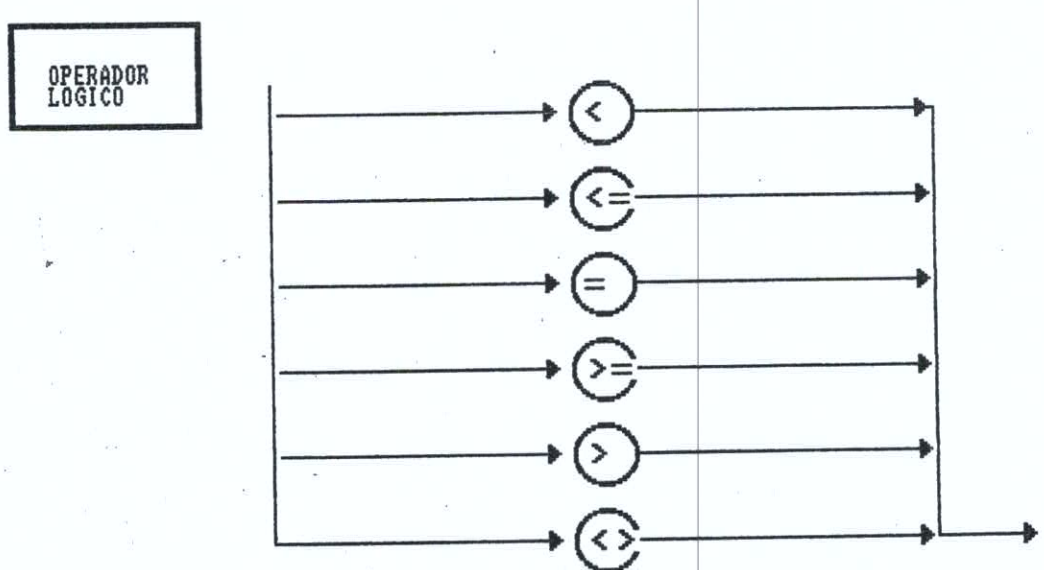

LISTA DE PARAMETROS
<
IDENTIFIC
=
TIPO
>
,
;

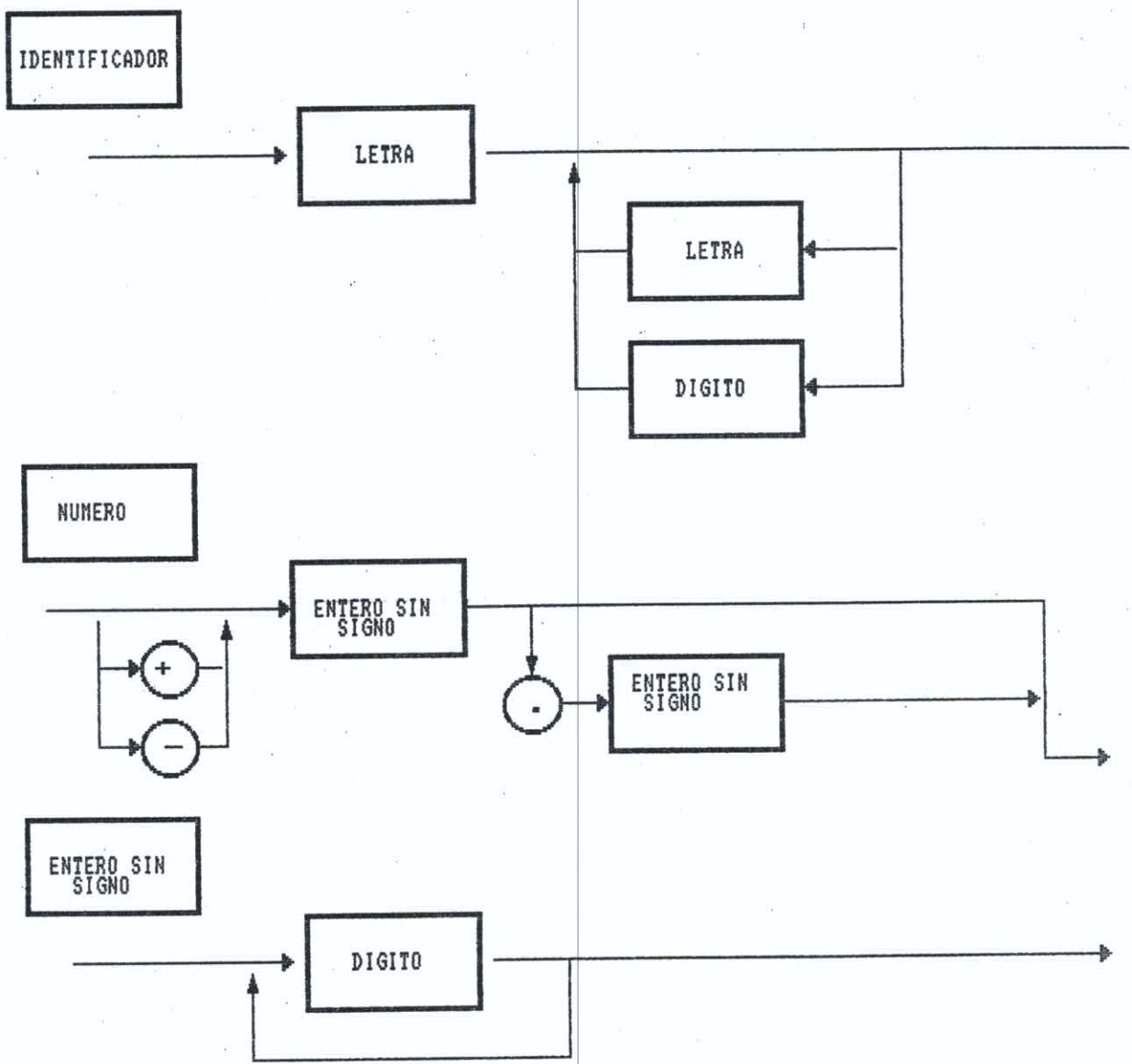

EXPRESION
TERMINO
+
-
TERMINO
TERMINO
FACTOR
*
/
FACTOR
FACTOR
(
EXPRESION
)
VARIABLE
CONSTANTE
IDENTIFICAD. DE FUNCION
(
EXPRESION
)
,


EXPRESION LOGICA
TERMINO LOGICO
O
TERMINO LOGICO


TERMINO LOGICO
FACTOR LOGICO
Y
FACTOR LOGICO


FACTOR LOGICO
(
EXPRESION LOGICA
)
VARIABLE
OPERADOR LOGICO
VARIABLE


OPERADOR LOGICO
<
<=
=
>=
>
<>


IDENTIFICADOR
LETRA
LETRA
DIGITO
NUMERO
ENTERO SIN SIGNO
+
-
-
ENTERO SIN SIGNO
ENTERO SIN SIGNO
DIGITO