TESIS

AJUSTE DE REGULADORES PID EN SISTEMAS MULTIVARIABLES

por

PEDRO OLLERO DE CASTRO

Ingeniero Industrial por la E.T.S. de I.I.
de la Universidad de Sevilla

presentada en la

ESCUELA TECNICA SUPERIOR DE INGENIEROS INDUSTRIALES

de la

UNIVERSIDAD DE SEVILLA

para la obtención del

Grado de Doctor Ingeniero Industrial

SEVILLA, OCTUBRE 1975

T E S I S

AJUSTE DE REGULADORES PID EN SISTEMAS MULTIVARIABLES

por

Pedro OLLERO DE CASTRO

Ingeniero Industrial por la E.T.S.de I.I.

de la Universidad de Sevilla

presentada en la

ESCUELA TECNICA SUPERIOR DE INGENIEROS INDUSTRIALES

de la

UNIVERSIDAD DE SEVILLA

para la obtención del

Grado de Doctor Ingeniero Industrial

Sevilla, Octubre de 1.975

TESIS DOCTORAL

AJUSTE DE REGULADORES PID EN SISTEMAS MULTIVARIABLES

Por : D. Pedro Ollero de Castro

Director de Tesis : Prof. Dr. Javier Aracil Santonja

TRIBUNAL CALIFICADOR

Presidente : Prof. Dr. Gabriel A. Ferraté Pascual
Universidad Politécnica de Barcelona

Vocales : Prof. Dr. Eugenio Andrés Puente
Universidad Politécnica de Madrid

Prof. Dr. Juan Manuel Martínez Moreno
Universidad de Sevilla

Prof. Dr. Antonio Martín Pérez
Universidad de Bilbao

Prof. Dr. Javier Aracil Santonja
Universidad de Sevilla

Sevilla, Octubre de 1.975

El autor expresa al Profesor Javier Aracil Santonja, bajo cuya dirección ha sido realizada esta tesis, su profundo y sincero agradecimiento por su continua dedicación, estímulos y consejos.

Igualmente agradece a D. Cayetano García - Montes, D. Eduardo Fernández Camacho y demás compañeros del Departamento de Control Automático, por la eficaz colaboración prestada durante el - desarrollo de este trabajo.

Asimismo manifiesta su agradecimiento a todo el personal adscrito al Centro de Cálculo de la Escuela Técnica Superior de Ingenieros Industriales por la inestimable ayuda prestada.

PLANTEAMIENTO Y RESUMEN DE LA TESIS

Se aborda en este trabajo, la obtención de un método de ajuste de reguladores industriales con aplicación a procesos multivariables.

Basicamente se presentan dos problemas cuando se pretende regular un proceso ,

a) la descripción matemática del comportamiento del proceso

b) el diseño del sistema de control que haga cumplir las especificaciones prefijadas.

Los procesos industriales suelen ser sistemas multivariables no lineales siendo precisamente esta clase de sistemas, la que más dificultades plantea al tratar de abordar tanto el problema de la identificación como el problema del diseño.

Frente a la complejidad de las técnicas de identificación desarrolladas (apartado 1.3), se estudia en este trabajo, la extensión a sistemas continuos del conocido método de identificación paramétrica - por mínimos cuadrados, aplicable en principio, a sistemas discretos.El empleo del algoritmo de transformación propuesto (apartado 2.), permite obtener un modelo linealizado del proceso. La validezdel método - queda patentizada por los excelentes resultados conseguidos en las aplicaciones realizadas (cap. V).

La obtención de un modelo linealizado del proceso hace factible abordar el problema del diseño mediante cualquiera de las técnicas desarrolladas para sistemas multivariables lineales.

Del estudio realizado sobre las técnicas de diseño empleadas hasta la fecha (apartado 1.4), cabe resaltar dos puntos en los que se pone de manifiesto la complejidad de sus aplicaciones a procesos industriales. Mientras más redundan en un órgano de control excesivamente complicado y costoso, otros presentan una gran dificultad operativa y experiencia en el diseño.

Teniendo presente estas limitaciones se estudia en los capítulos III y IV una técnica de ajuste de reguladores industriales. Aunque necesita de un ordenador para su desarrollo, la complejidad operativa no es excesiva, permitiendo rapidez de cálculo sin apenas experiencia previa en el diseño. Los resultados obtenidos en las aplicaciones realizadas (apartado 5. y 5), garantizan la validez del método propuesto.

INDICE

Capitulo I.- INTRODUCCION.

1.1. Antecedentes históricos..... 1

1.2. Definiciones y conceptos básicos..... 2

1.3. El problema de la identificación de sistemas..... 7

1.4. El problema del diseño de sistemas de control.....10

Capitulo II.- IDENTIFICACION DE MODELOS PARAMETRICOS EN SISTEMAS MULTIVARIABLES. TRANSFORMACION DE UN MODELO DISCRETO A UN MODELO CONTINUO.

2.1. Estimación de parametros por minimos cuadrados 17

2.1.1. Determinación del orden 20

2.1.2. Algoritmo recurrente para el calculo en un computador digital 21

2.1.3. Extensión a sistemas multivariables 24

2.2. Transformación de un modelo discreto a un modelo continuo 27

2.2.1. Estudio de las ecuaciones de tranformación 31

2.2.2. Algoritmo de transformación 33

2.2.3. Elección de las señales de entrada 45

Capitulo III.- METODO DE AJUSTE DE REGULADORES PID PARA SISTEMAS UNA ENTRADA-UNA SALIDA.

3.1. Métodos elasticos de diseño 47
3.1.1. Planteamiento general del problema 47
3.1.2. Problema del seguimiento 49
3.1.3. Problema del regulador 50
3.2. Método de ajuste de reguladores PID 51
3.2.1. Fundamentos del método propuesto 51
3.2.2. Reguladores P, PI, PID 54
3.2.3. Comportamiento deseado en bucle cerrado ... 57
3.2.4. Descomposición en fracciones continuadas .. 59
3.2.5. Desarrollo del método de ajuste 61

Capitulo IV.- METODO DE AJUSTE DE REGULADORES PID PARA SISTEMAS MULTIVARIABLES.

4.1. Planteamiento general del problema 73
4.2. Indice de inestabilidad estructural 79
4.3. Elección de los bucles de control .....
4.4. Estabilidad de sistemas multivariables 84
4.5. Ajuste de reguladores PID 89
4.5.1. Algoritmo de ajuste 97

Capitulo V.- APLICACIONES.

5.1. Aplicaciones del método de identificación 100
5.1.1. Sistema 2 x 2 con polos reales 100

5.1.2. Sistema 2 x 2 con polos complejos 112

5.2. Ajuste de reguladores PID en sistemas monovariables. 124

5.2.1. Motor de corriente continua alimentado por rectificadores estáticos 124

5.2.2. Vaporizados LPG 131

5.3. Ajuste de reguladores PID en sistemas multivariables 143

5.3.1. Mezcladores de tanque agitado 143

5.3.2. Turbina de gas 152

5.3.3. Sistema 3 x 3 176

CONCLUSIONES 181

REFERENCIAS 183

Apendice A

A.1. Cálculo de los autovalores de la matriz A de un sistema continuo a partir de los autovalores de la matriz P del sistema discreto correspondiente 191

A.2. Acotación de errores en el desarrollo de la exponencial matricial 194

Apéndice B

B.1. Comportamiento deseado en bucle cerrado. Sistemas seguidores 199

B.2. Comportamiento deseado en bucle cerrado. Sistemas reguladores 202

Apéndice C

C.1. Programas FORTRAN para el método de Identificación descrito en el Capítulo II 206

C.2. Programa FORTRAN para el método de ajuste de reguladores PID descrito en el Capítulo III.. 224

C.3. Programa FORTRAN para el método de ajuste de reguladores PID descrito en el Capítulo IV.. 224

C.4. Programa FORTRAN correspondiente a las aplicaciones realizadas 257

CAPITULO I

INTRODUCCION.-

1.1.- Antecedentes históricos.

Aunque ya a mediados del siglo XVIII aparecen los primeros sistemas automáticos de control, éstos no son objeto de tratamiento matemático hasta aproximadamente un siglo después (1868), año en el que Maxwell - (ref.1), publica el primer trabajo sobre un mecanismo de control.

Durante las dos décadas precedentes a la Segunda Guerra Mundial, coincidiendo con el gran desarrollo de la teoría de circuitos, Nyquist - (ref. 2) y Black (ref.3), publican los primeros trabajos sobre estabilidad de sistemas lineales realimentados. Estos dos trabajos serán el origen de la Teoría de Control que no tardaria en desarrollarse. Es en 1942 cuando Harris (ref.4), introduce el concepto de "función de transferencia", cuya utilización constituye la base de la que se ha dado en llamar "Teoría Clásica de Control". Bode (ref.5), logra ya importantes avances en la teoría de Control y ataca el problema del diseño.

La aparición poco después, de los métodos sobre el lugar de las raices publicados en 1948 por Evans (ref. 6), constituyen un nuevo enfoque y una nueva forma de diseño, la síntesis directa.

La teoría clásica de control presenta sin embargo grandes limitaciones en el tratamiento de problemas, como "control óptimo" o diseño de sistemas de control multivariables. Es por esto, por lo que a mediados de la década de los cincuenta se abre una nueva línea de investigación en la teoría de control. Esta nueva línea utiliza el dominio del tiempo y la descripción interna en la modelización del sistema. - Existe una gran diferencia conceptual entre ambas técnicas como lo ponen de manifiesto Wiener (ref.7) y Kalman (ref.8) al tratar el problema del filtrado óptimo.

La Teoría Moderna del Control profundiza más en el concepto matemático del problema. Así Kalman (ref.9), define los conceptos de controlabilidad y observabilidad, mediante la formulación por variables - de estado.

El problema de diseño de sistemas de control multivariables lineales e invariantes en el tiempo fue un campo rápidamente invadido por la Teoría Moderna de Control. Sin embargo, recientemente Rosenbrock (ref.10), ha conseguido extender los métodos clásicos al diseño de sistemas multivariables.

1.2.- Definiciones y conceptos básicos.

Debido a que en la teoría del control automático algunos términos poseen diversas acepciones, es interesante dar a conocer la inter -

pretación que se hará de ellos a lo largo del presente trabajo.

1.2.1.- Concepto de Sistema.

De un modo general puede considerarse un "sistema" como un conjunto de elementos interrelacionados entre sí, de forma que ciertas magnitudes asociadas a dichos elementos, variables del sistema, pueden influir o ser influidos por las demás. Esta definición puede completarse mediante una clasificación de las variables del sistema y una representación simbólica del mismo, fig. 1.1.

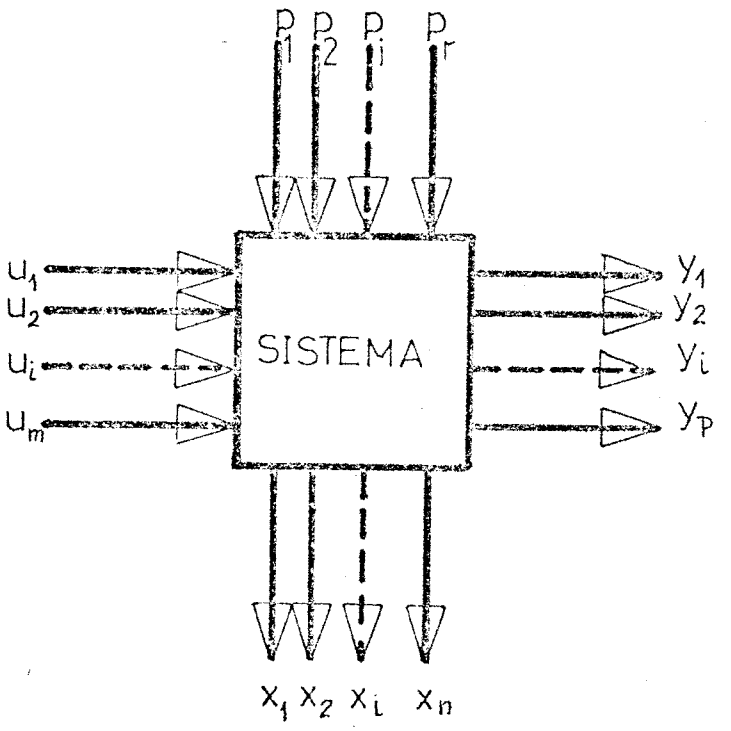


fig. 1.1.

En la fig. 1.1, pueden observarse dos tipos de variables que actuan sobre el sistema. Las variables u_i se denominan variables de entrada ó variables manipuladas ó de control. El valor de estas variables puede ser modificado voluntariamente desde el exterior. Las variables p_i se denominan variables de perturbación. No es posible actuar sobre estas variables, que sólo en algunos casos son fisicamente medibles.

Por otra parte existen dos grupos de variables que informan de la evolución del sistema. Las variables y_i son denominadas variables de salida ó variables controladas . Por último, el conjunto de variables x_i es el conjunto mínimo de variables del sistema, tal que, conocido en un instante dado, permite conocer la evolución del sistema a partir de ese instante ante cualquier señal de entrada o de perturbación. Este conjunto de variables se denomina vector de estado.

Un sistema dinámico es un modelo matemático de un sistema físico, que en teoria de control ha sido representado de dos formas distintas, la descripción interna y la descripción externa.

1.2.2.- Descripción externa.

La descripción externa de un sistema es una representación matemática del tipo :

$$\bar{y}(t) = f\left[\bar{u}(t_o, t)\right]$$

donde f es por consiguiente una función que relaciona directamente los vectores de entrada y salida, haciendo corresponder al conjunto de valores tomados por el vector de entrada $\bar{u}(t)$, en el intérvalo (t_o, t) el valor tomado por el vector de salida en el instante t.

Normalmente se emplean dos formas de descripción externa : la respuesta impulsional y la matriz de transferencia. En la práctica cuando el sistema objeto de estudio es lineal invariante en el tiempo y poseee condiciones iniciales nulas se utiliza la matriz de transferencia como forma de descripción externa. En este caso el sistema se representa en ausencia de perturbaciones por

$$\bar{y}(s) = G(s)\ \bar{u}(s)$$

donde $\bar{y}(s)$ y $\bar{u}(s)$ son los vectores transformados por Laplace de los vectores de salida y entrada respectivamente. Si el sistema posee m entradas y p salidas, la dimensión de la matriz de transferencia será $p \times m$. Así por ejemplo, el elemento (i,j) de la matriz T(s) será una función racional en s que representa el cociente entre la transformada de Laplace de la variable de salida i-ésima y la transformada de Laplace de la señal de entrada j-ésima.

Si el sistema es de una entrada - una salida, G(s) es ahora un escalar que se denomina función de transferencia.

1.2.3.- Descripción interna.

La descripción interna establece una relación explícita indirecta entre las variables de entrada y salida. Formalmente puede escribirse :

$$\dot{\bar{x}}(t) = f[\bar{x}(t), \bar{u}(t), \bar{p}(t), t] \qquad (1.3)$$

$$\bar{y}(t) = g[\bar{x}(t), \bar{u}(t), \bar{p}(t), t] \qquad (1.4)$$

donde f y g se denominan función de transición y función de lectura.

Para la clase de sistemas lineales e invariantes en el tiempo la descripción interna admite una representación matemática en la forma

$$\dot{\bar{x}}(t) = A\,\bar{x}(t) + B\,\bar{u}(t) + E\,\bar{p}(t) \qquad (1.5)$$

$$\bar{y}(t) = C\,\bar{x}(t) + D\,\bar{u}(t) + F\,\bar{p}(t) \qquad (1.6)$$

donde para un sistema con m entradas, r perturbaciones, p salidas y n elementos en el vector de estados, A,B,E,C,D y F son matrices de dimensiones $n \times n$, $n \times m$, $n \times r$, $p \times n$, $p \times m$, y $p \times r$ respectivamente.

3.1.- El problema de la identificación de sistemas.

El primer problema que cronológicamente debe abordarse en el estudio de un sistema es el de su identificación. De acuerdo con la definición realizada, un sistema dinámico se caracteriza por relaciones funcionales entre sus variables de entrada y salida. Se trata, pues, en primer lugar, de saber el tipo de relación funcional existente y plasmar este conocimiento en la consecución del modelo matemático del sistema. Bien entendido que no bastará con saber el tipo de relación y que habrá que cuantificarla, determinando lo que se ha dado en llamar "parámetros del sistema". Si se conocen las leyes físicas involucradas en el sistemas, será sencillo establecer el modelo del sistema. Sin embargo es corriente, que las relaciones entre variables del sistema sean tan complejas, que no permitan una descripción matemática sencilla. Habrá que recurrir en este caso a un método experimental excitando el sistema con diversas señales de entrada normalizadas y observando la respuesta del mismo.

Matemáticamente el problema consiste en determinar los parámetros $q(t)$, que caracterizan la ecuación diferencial vectorial

$$\dot{\bar{x}}(t) = f\left[\bar{x}(t), \bar{u}(t), \bar{q}(t), \bar{w}(t)\right] \qquad (1.6)$$

$$\bar{y}(t) = g\left[\bar{x}(t), \bar{u}(t), t\right] + h\left[\bar{x}(t), \bar{u}(t), \bar{v}(t)\right]$$

donde :

$x(t)$ = vector de estados

$u(t)$ = vector de mando

$q(t)$ = vector de parámetros desconocidos

$w(t)$ = vector de ruido a la entrada

$v(t)$ = vector de errores en la medida

$y(t)$ = vector de variables de salida

Gran parte de los sistemas físicos pueden ser considerados como aproximadamente lineales ó bien, pueden ser linealizados a lo largo de una trayectoria nominal. En ese caso (1.6) puede ser escrito en la forma

$$\dot{\bar{x}}(t) = A(t)\,\bar{x}(t) + B(t) + E(t)\,\bar{w}(t)$$

$$\bar{y}(t) = C(t)\,\bar{x}(t) + D(t)\;\; v(t) \qquad (1.7)$$

El objeto de la identificación de sistemas lineales es entonces determinar las matrices involucradas en (1.7). Si se considera además, que los parámetros del sistema no varian con el tiempo o lo hacen de forma muy lenta, las matrices antes consideradas tienen elementos constantes, pudiéndose por otra parte acudir a la representación del sistema en el dominio de la frecuencia mediante las funciones o matrices de transferencia. Se entra así, en la identificación de sistemas lineales invariantes en el tiempo.

Han sido numerosos los métodos desarrollados al abordar el problema de la identificación para esta clase de sistemas. Las técnicas de correlación, cuyas propiedades y aplicación son bien conocidas, han sido estudiadas recientemente por Rake(ref.11), Welfonder (ref. 12) , Buchta (ref. 13), Reid(ref.14) Gerdin (ref.15), Stassen (ref.16) y otros.

El análisis espectral basado en la aplicación de la transformada de Fourier, determina la función de transferencia a partir de las funciones de densidad espectral de la entrada y de la salida. Una buena descripción de esta técnica ha sido publicada por Jenkins (ref. 17).

La deconvolución numérica es otro método de identificación en el que se determina la respuesta impulsional mediante la transformada inversa de la función racional definida por $G(s) = y(s) / u(s)$.

Algunos trabajos realizados sobre este método son los de Stanley (ref. 18), Fielder (ref. 19), y Depeyrot (ref. 20).

No hay que olvidar en el conjunto de técnicas de identificación el algoritmo de Ho (ref. 21), cuya aplicación a sistemas discretos dá óptimos resultados.

La técnica, quizás más utilizada durante los últimos años, es la de identificación de modelos paramétricos por mínimos cuadrados . Esta técnica que se desarrollará más profundamente en un capítulo - posterior presenta indudables ventajas sobre las anteriores :

- es fácil minimizar un criterio de error entre la salida real y la del modelo identificado.
- la consideración de señales de ruido no plantea grandes dificultades.
- el proceso de cálculo es de sencilla implementación en un computador.
- la extensión a sistemas multivariables es directa.

Frente a esta serie de ventajas un único inconveniente desde el punto de vista del diseño de un sistema de control ; el modelo que se obtiene por este procedimiento es discreto, es decir, basado en la transformada z.

Algunos de los más importantes trabajos realizados en este campo son los de Mann y Wald (ref. 22), Astrom,(ref.23) y Clarke (ref.24).

3.2.- El problema de diseño de sistemas de control

El problema del diseño consiste en determinar una configuración de elementos u órganos de control de forma, que acopladas al sistema fí-

sico formen un sistema dinámico que cumpla unas determinadas especifi caciones.

En función del tipo de especificación o propiedades exigidas al sistema, el problema de diseño puede ser dividido en tres subpro - blemas.

a) Problema del regulador : que consiste en determinar un sistema de control que mantenga insensible la salida frente a la acción de - perturbaciones exteriores.

b) Problema de seguimiento : lograr que el sistema siga con preci sión la evolución de una señal o variable de referencia.

c) Problema del control óptimo : determinar un sistema de control que cumpla unos objetivos prefijados maximizando o minimizando un índice de calidad.

En cualquiera de sus tres acepciones, el problema de diseño de sistemas de control lineales invariantes en el tiempo está resuelto pa ra el caso una entrada-una salida. No ocurre lo mismo para el caso - multivariable en el que, a pesar de los numerosos trabajos de investiga ción realizados, no se ha llegado a una solución total del problema . Hammond (ref. 25), y más recientemente Mac Farlane (ref.26), ponen de ma nifiesto esta situación en dos trabajos de recopilación que totalizan más de 600 referencias.

Antes de pasar a una somera descripción de las técnicas de diseño, actualmente en boga para sistemas multivariables, conviene resaltar ciertos aspectos del problema.

Una cuestión llave es la siguiente : ¿ por qué no diseñar separada y sucesivamente cada uno de los bucles de realimentación considerado como sistema una entrada una salida ? La razón está en que el sistema padece de interacciones entre variables. Desde el punto de vista de la Teoría Clásica, basándose en la respuesta en frecuencia, puede decirse que la interacción entre variables de distintos bucles ocasiona normalmente una reducción en las márgenes de estabilidad del sistema. Es pués necesario tener en cuenta en el diseño, el efecto producido por las interacciones.

Diferentes formas de atacar el problema de las interacciones han dado lugar a distintos métodos de diseño :

1) Diseño por desacoplamiento . Si se adopta el esquema de control representado por el diagrama de bloques de la figura 1.2 se trata de determinar una matriz de compensación K(s) de forma que G(s) K(s) sea diagonal. El sistema resultante se dice que está desacoplado, no hay interacción entre pares de variables entrada-salida. El diseño es entonces fácil y se reduce a aplicar las técnicas conocidas por una entrada-una salida. Este método propuesto por Boksenbom y Hood (ref. 27), presenta graves inconvenientes como lo pone de manifiesto Rae (ref. 28);

a) da lugar la mayoria de las veces a una matriz de compensación no realizable fisicamente o excesivamente compleja;

b) gran parte de la libertad en el diseño se consume en desacoplar el sistema, lo que da lugar a un pobre comportamiento dinámico ;

c) si det | G(s)| presenta polos con parte real positiva el método puede dar lugar a un sistema inestable.

II) Control por asignación de polos (Modal control). Esta técnica basada en la realimentación lineal del estado, permite situar los polos del sistema en bucle cerrado en cualquier situación deseada. Sin embargo no se consigue ningún control sobre los ceros, por lo que no es posible ajustar correctamente la respuesta transitoria. Por otra parte, dado que en realidad no se controla el estado, sino sólo una combinación lineal de las variables de estado, en algunos casos el diseño efectuado no responde adecuadamente frente a perturbaciones. Por último, en el caso en que las variables de estado no sean accesibles se hace necesario el diseño de un observador, lo cual complica la realización física del órgano de control.

III) Técnica del control conmutativo : Es un método que corresponde a un tratamiento completamente algebraico del problema de diseño. Básicamente consiste en exigir para determinadas frecuencias ciertas propiedades a los módulos de los autovalores de la matriz de transferencia en bucle abierto G(s). K(s). El punto crucial del método , radica en la elección de los elementos de K(s) de forma que se cum -

plan las propiedades deseadas en los autovalores de la matriz producto. Esto constituye un problema complejo, ya que en la actualidad, poco se conoce sobre la situación de los autovalores de una matriz producto en función de la situación de los autovalores de las matrices que se multiplican. Mac Farlane (ref.29) y (ref.30), apunta la idea de hacer mediante una transformación en las bases del espacio vectorial, que las dos matrices conmuten, teniendo por tanto el mismo conjunto de autovalores. El método se reduce entonces al diseño de m bucles una entrada-una salida. La complejidad de cálculo, que hace inviable el método en muchos casos, es debida a que las funciones de transferencia características de G(s) son irracionales.

IV) Técnica del lugar característico. Empleando el lugar característico, representación de la respuesta en frecuencia de los autovalores de la matriz de transferencia de un sistema, Mac Farlane y Belletritti (ref.31), han desarrollado una técnica de diseño que permite conocer exactamente el grado de estabilidad del sistema multivariable en bucle cerrado. Este método necesita de un computador con salida gráfica y de experiencia en el diseño.

V) Técnica del lugar inverso de Nyquist . Es quizás la técnica más potente en el tratamiento del problema de diseño para sistemas multivariables. Fue introducida por Rosenbrock (refs. 10,32,33,34) que creó su fundamento enunciando el "teorema de estabilidad de Rosenbrock" y aplicando por primera vez los conceptos "dominancia diagonal" en las matrices de transferencia. El método necesita de la representación gráfica

por computador de los lugares inversos de Nyquist correspondientes a los elementos diagonales de la matriz de transferencia en bucle abierto. Sobre esta representación gráfica se estudian las condiciones de estabilidad y dominancia diagonal. Caso de que no se cumpla alguna de esta propiedades se diseña una matriz de precompensación. El resto del diseño puede realizarse considerando m bucles simples de una entrada-una salida ya que la dominancia diagonal asegura que el efecto de las interacciones es muy limitado.

Siguiendo la misma línea, Mayne (ref. 35) ha desarrollado un método que utiliza la representación gráfica del lugar directo de Nyquist, realizando al mismo tiempo un estudio de estabilidad en caso de ruptura en alguna de las cadenas de realimentación.

VI) Control óptimo. El problema del diseño se enfoca en este caso bajo una óptica distinta. Se trata de conseguir unos objetivos en las variables de salida optimizando un índice de calidad. Esta técnica desarrollada por muchos autores, Kalman (ref.36), Atans, y Falb (ref.37), Tyler (ref.38) y otros, presenta indudables ventajas junto a graves inconvenientes. Entre las primeras, el llegar a conocer una solución explícita y el poder utilizar índices de rendimiento económico. Por el contrario se precisa de un modelo matemático muy preciso y de la accesibilidad de todas las variables de estado. Además en bastantes casos prácticos es dificil elegir un indice de calidad apropiado.

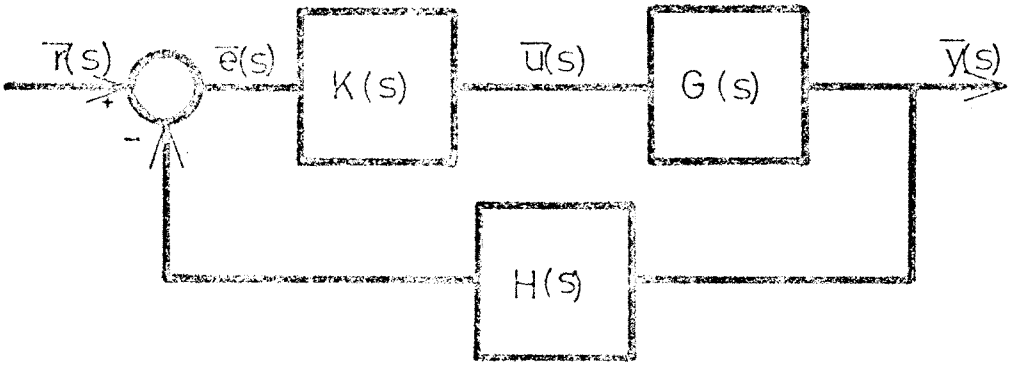


fig. 12

CAPITULO II

IDENTIFICACION DE MODELOS PARAMETRICOS EN SISTEMAS MULTIVARIABLES. TRANSFORMACION DE UN MODELO DISCRETO A UN MODELO CONTINUO.

Se describe en este capítulo la obtención de los parámetros de la matriz de transferencia de un sistema continuo lineal e invariante en el tiempo. Excitando al sistema con entradas de tipo escalón y muestreando la respuesta se obtiene, aplicando el método de ajuste por mínimos cuadrados, el modelo correspondiente a un sistema discreto ficticio. El método de transformación propuesto, permite obtener finalmente la matriz de transferencia correspondiente al sistema real.

2.1.- Estimación de parámetros por mínimos cuadrados.

Considérese el modelo discretizado correspondiente a un sistema lineal invariante en el tiempo de una entrada - una salida, representado por la ecuación en diferencias

$$y_m(k) + a_1\ y_m(k-1) + \ldots + a_n\ y_m(k-n) = b_1\ u(k-1) + \ldots b_n\ u(k-n) \qquad (2.1)$$

o por la función de transferencia pulsional

$$G(Z) = \frac{b_1\ Z^{-1} + b_2\ Z^{-2} + \ldots\ b_n\ Z^{-n}}{1 + a_1\ Z^{-1} + \ldots\ a_n\ Z^{-n}} = \frac{B(Z^{-1})}{A(Z^{-1})} \qquad (2.2.)$$

Se establece la función de error según la expresión :

$$V = V(y, y_m) = \sum_{k=0}^{k+n} e^2(k) \qquad (2.3)$$

donde e(k) es el error generalizado

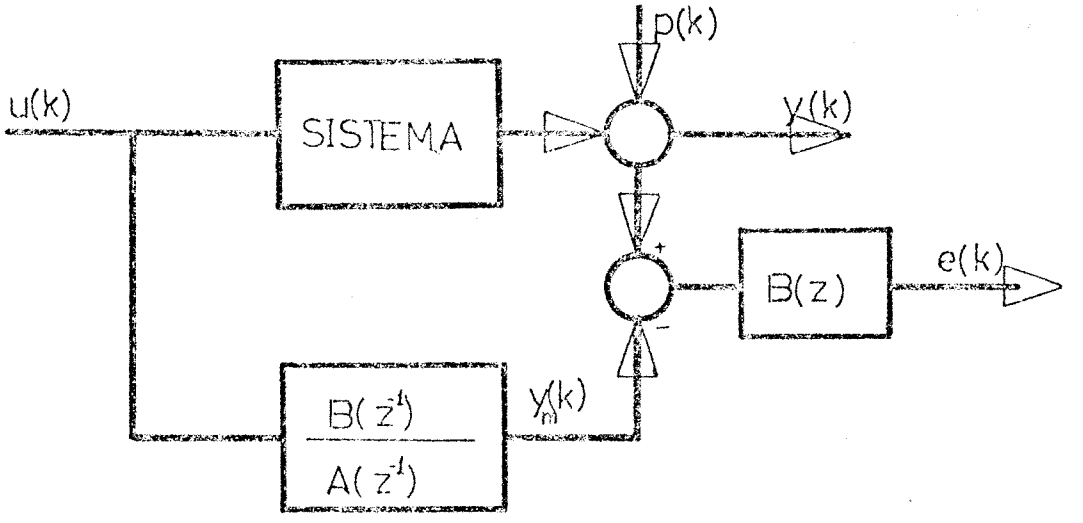
$$e(k) = A(Z^{-1}) \left[y(k) - y_m(k) \right] \qquad (2.4)$$

que también puede ser escrito en la forma

$$e(k) = A(Z^{-1})\ y(k) - B(Z^{-1})\ u(k) \qquad (2.5)$$

Definido de esta forma, el error es una función lineal en los parámetros a_i y b_i y la función de error es cuadrática, resultando - por tanto fácil determinar analíticamente el mínimo.

En la figura (2.1) se muestra una descripción simbólica de la ecuación de error (2.4).



Nótese que (2.5) implica que :

$$y(k) + a_1\, y(k-1) + \ldots + a_n\, y(k-n) = b_1\, u(k-1) + \ldots\ b_n(k-n) + e(k) \tag{2.6}$$

Adoptando una representación matricial, (2.1) puede ser escrito en la forma :

$$Y_k = \Phi_k \cdot \beta_k \tag{2.7}$$

$$\Phi_k = \left[\begin{array}{cc:c} -y(0) & -y(-1)\ldots -y(1-n) & u(0)\ u(-1)\ldots u(1-n) \\ -y(1) & -y(0) & u(1)\ u(0)\ \ldots\ u(2-n) \\ \vdots & & \vdots\ \ \vdots \\ -y(k-1) & -y(k-2)\ldots -y(k-n) & u(k-1)\ldots\ldots u(k-n) \end{array}\right] \tag{2.8}$$

$$\beta_k^T = \left[a_1,\ a_2 \ldots\ a_n\ \ b_1,\ b_2 \ldots\ b_n\right]$$

La ecuación de error (2.5) es entonces :

$$e = Y_k - \Phi_k \beta_k \tag{2.9}$$

y el mínimo de la función de error se determina a partir de $\nabla V = 0$. Si la matriz $\Phi_K \Phi_K^T$ es no singular, el mínimo de V se obtiene (ref.39) para

$$\hat{\beta} = \Phi_K^+ Y_K \tag{2.10}$$

donde Φ^+ es la pseudo inversa de Φ definida por

$$\Phi_K^+ = \left[\Phi_K^T \Phi_K\right]^{-1} \Phi^T \tag{2.11}$$

2.1.1.- Determinación del orden.

En la práctica, pocas veces es conocido a priori el orden del sistema con el agravante, de que pueden cometerse grandes errores cuando la estimación de éste ha sido incorrecta.

Para determinar el orden del sistema se puede proceder a aplicar el algoritmo anterior, varias veces, con diferentes órdenes, hasta que la disminución de la función de error sea despreciable. Con el objeto de cuantificar esta reducción Anderson(ref. 40), utiliza el índice t

$$t = \frac{V_1 - V_2}{V_2} \quad \frac{k - n_2}{n_2 - n_1} \tag{2.12}$$

donde V_i es el valor mínimo de la función de error cuando el modelo tiene n_i parámetros y k es el número de pares de valores entrada - salida. Apoyándose en el hecho de que $(n_2 - n_1)\, t$ presenta una - distribución χ^2, Anderson muestra que $t > 3$, para que la reducción en la función de error sea significativa.

Otra forma de determinar el orden del sistema ha sido propuesta por Sinha (ref. 41), para el caso de que no existan errores en la medida, esto es $\rho(k) = 0$ en la fig. 2.1.

2.1.2.- Algoritmo recurrente para el cálculo en un computador digital.

La aplicación directa de (2.10) presenta el inconveniente de tener que rehacer todo el cálculo para cada nuevo par de valores entrada - salida, ya que hay que considerar en cada una de las matrices una fila más. Por otra parte, si las operaciones se realizan en un - computador, la memoria ocupada aumenta continuamente, conforme se van tomando nuevas muestras del proceso.

Por este motivo y considerando además la posibilidad de identificación en línea, Sinha (ref. 41), basándose en los trabajos de Greville (ref. 42) y Albert (ref.43) ha desarrollado el siguiente algoritmo recurrente,

Sea :

$$Y_{k+1}^{T} = \left[Y_k \quad y(k+1) \right]$$

$$\Phi_{k+1} = \begin{bmatrix} \Phi_K \\ \partial_{k+1}^{T} \end{bmatrix} \tag{2.13}$$

y $\partial_{k+1}^{T} = \Big[-y(k)\ y(k-1) \ldots y(k-n-1)\ u(k)\ u(k-1) \ldots$

$\ldots \quad u(k-n-1) \Big]$

se tiene entonces que :

a) para $k < 2n$

$$\hat{\beta}_{k+1} = \hat{\beta}_k + \frac{(Q_k \cdot \partial_{k+1})}{\partial_{k+1}^{T} \cdot Q_k \cdot \partial_{k+1}} \left(y(k+1) - \partial_{k+1}^{T} \cdot \hat{\beta}_k \right)$$

donde :

$$Q_{k+1} = Q_k - \frac{(Q_k \cdot \partial_{k+1})\,(Q_k \cdot \partial_{k+1})^{T}}{\partial_{k+1}^{T} \cdot Q_k \cdot \partial_{k+1}} \tag{2.14}$$

y

$$P_{k+1} = P_k - \frac{(P_k\,\partial_{k+1})\,(P_k\,\partial_{k+1})^T + (Q_k\,\partial_{k+1})\,(P_k\,\partial_{k+1})^T}{\partial_{k+1}^T \cdot Q_k \cdot \partial_{k+1}}$$

$$\frac{(Q_k\,\partial_{k+1})\,(Q_k\,\partial_{k+1})^T\,(1 + \partial_{k+1}\,P_k\,\partial_{k+1})}{(\partial_{k+1}^T \cdot Q_k \cdot \partial_{k+1})^2}$$

con las condiciones iniciales

$$Q_0 = 1 \qquad P_0 = 0 \qquad \beta_0 = 0$$

b) para $k > 2n$

$$\hat{\beta}_{k+1} = \hat{\beta}_k + \frac{P_k\,\partial_{k+1}\,(y(k+1) - \partial_{k+1}^T\,\hat{\beta}_k)}{1 + \partial_{k+1}^T \cdot P_k \cdot \partial_{k+1}}$$

donde : (2.15)

$$P_{k+1} = P_k - \frac{P_k\,\partial_{k+1}\,(P_k\,\partial_{k+1})^T}{1 + \partial_{k+1}^T\,P_k\,\partial_{k+1}}$$

En el apéndice (C) se muestra el programa Julio, versión ligeramente modificada del realizado por G. Montes (ref.44), que desarrolla los cálculos del algoritmo recurrente según dos opciones. Mientras una determina β_k para $k = 2n$ mediante la expresión $\beta_{2n} = \Phi \cdot Y_{2n}$, la otra comienza desde la determinación de β_1.

2.1.3.- Extensión a sistemas discretos multivariables.

La extensión del método de identificación por mínimos cuadrados a sistemas multivariables discretos (ref.45) es directa, si el sistema se descompone de la siguiente forma para cada canal de salida :

$$y_i(k) = -a_i\, y_i(k-1) \ldots -a_n\, y_i(k-n) + b_1^{i1}\, u_1(k-1) + b_n^{i1}\, u_1(k-n) + \ldots + b_1^{ip}\, u_p(k-1) + b_n^{ip}\, u_p(k-n)$$

para $i = 1, m$ (2.16)

donde :

$y_i(k)$ = respuesta del canal i en el instante de muestreo k

m = número de salidas

p = número de entradas

Más concisamente (2.16) puede escribirse en la forma :

$$y_i(k) = -\sum_{j=1}^{n} a_j\, y_i(k-j) + \sum_{r=1}^{p} \sum_{j=1}^{n} b_j^{ir}\, u_r(k-j) \qquad (2.17)$$

Otra representación válida del sistema descrito en (2.16) o (2.17) corresponde a la siguiente matriz de transferencia en Z.

$$G(z) = \frac{1}{1+a_1 z^{-1}+\dots a_n z^{-n}} \begin{bmatrix} b_1^{11} z^{-1} + \dots b_n^{11} z^{-n} \dots b_1^{1p} z^{-1} + \dots \\ \vdots \qquad\qquad\qquad\qquad \vdots \\ b_1^{i1} z^{-1} + \dots b_n^{i1} z^{-n} \dots b_1^{ip} z^{-1} + \dots \\ \vdots \\ b_1^{n1} z-1+ \dots b_n^{m1} z^{-n} \dots b_1^{mp} z^{-1} + \\ + \dots b_n^{1p} z^{-n} \\ + \dots b_n^{ip} z^{-n} \\ + \dots b_n^{mp} z^{-n} \end{bmatrix} \qquad (2.18)$$

Para esta clase de sistemas el algoritmo recurrente descrito en el apartado anterior es válido, siempre que se adopte la siguiente representación matricial

$$Y_{ik}^{T} = \left[y_i(1)\; y_i(2) \dots y_i(k) \right]$$

$$\Phi_{ik} = \begin{bmatrix} -y_i(0)\ldots\ y_i(1-n) & u_1(0)\ldots\ u_1(1-n) & \ldots \\ -y_i(0)\ldots -y_i(2-n) & u_1(1)\ldots\ u_1(2-n) & \ldots \\ \vdots & & \\ -y_i(k-1)\ldots\ -y_i(k-n) & u_1(k-1)\ldots\ u_1(k-n) & \ldots \end{bmatrix}$$

$$\begin{matrix} \ldots & u_p(0)\ldots\ u_p(1-n) \\ \ldots & u_p(1)\ldots\ u_p(2-n) \\ & \vdots \qquad\quad \vdots \\ \ldots & u_p(k-1)\ldots\ u_p(k-n) \end{matrix} \Bigg] \qquad (2.19)$$

$$\hat{\beta}^T_{ik} = \begin{bmatrix} a_1 & a_2\ldots\ a_n & b_1^{i1} & b_2^{i1}\ldots\ b_n^{i1}\ldots\ b_1^{ip} & b_2^{ip}\ldots\ b_3^{ip} \end{bmatrix}$$

Esta descomposición permite obtener para cada aplicación del algoritmo recurrente una fila de la matriz de transferencia (2.18) y es adaptable fácilmente a una identificación en línea.

2.2.- Transformación de un modelo discreto a un modelo continuo.

En los apartados anteriores se ha mostrado como se obtiene un modelo discretizado de un sistema continuo, a partir de muestreos efectuados en las señales de entrada y salida. Se trata ahora de obtener las ecuaciones diferenciales que rigen el comportamiento del sistema a partir de las ecuaciones en diferencias cuyos parámetros han sido identificados. Es decir, en pocas palabras, el problema consiste ahora en obtener el modelo en "tiempo continuo" del sistema a partir del modelo discretizado. En algunos trabajos (ref. 46), se define este problema como "paso del dominio z al dominio s".

La fig. (2.2) representa un planteamiento esquemático de la situación a resolver

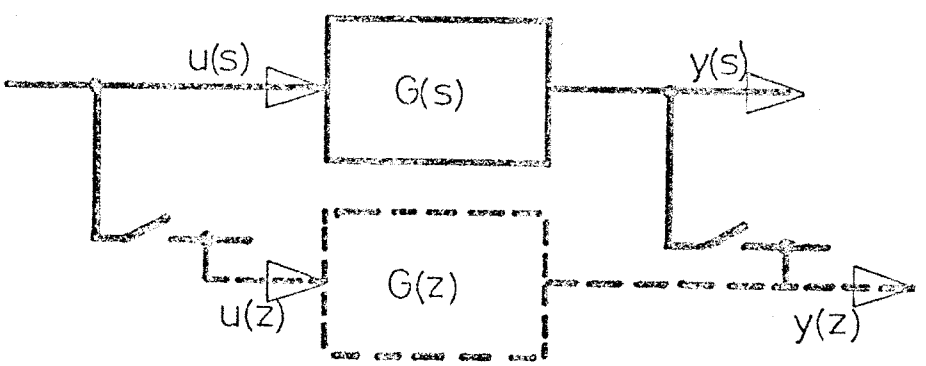


fig. 2.2

El problema de obtener G(s) a partir de G(z) no es posible abordarlo directamente. Sin embargo, es factible como se verá posteriormente, pasar de una descripción interna del sistema discreto definida por el triple [P,Q,C] a una descripción interna del sistema continuo definida por el triple [A, B, C] ; supuesto superado este paso, el obtener T(s) es trivial ya que

$$G(s) = C \ (sI - A)^{-1} B$$

En virtud de esto, el problema de transformación queda concretado a lo siguiente : obtenida una realización del sistema discreto de p entradas y m salidas :

$$x(k+1) = P\,x(k) + Q\,u(k)$$
$$y(k+1) = C\,x(k+1) \qquad (2.20)$$

donde :

$x(k+1)$ = vector de estados en el instante $(k+1)\cdot\Delta t$ de dimensión n.

$x(k)$ = vector de estados en el instante $k\Delta t$.

$y(k+1)$ = vector de salida en el instante $(k+1)\cdot\Delta t$ de dimensión m.

$u(k)$ = vector de entrada en el instante $k \cdot \Delta t$ de dimensión $p \times 1$

P = matriz dinámica de dimensión $n \times n$

Q = matriz de mando de dimensión $n \times p$

C = matriz de observación de dimensión $m \times n$

Δt = intervalo de muestreo.

se trata de determinar una realización del sistema en tiempo continuo tal como :

$$\begin{aligned} \dot{x}(t) &= A\,x(t) + B\,u(t) \\ y(t) &= C\,x(t) \end{aligned} \qquad (2.21)$$

donde la simbología utilizada es ya conocida y las dimensiones de las matrices integradas son idénticas a las de (2.20).

El problema así planteado, no siempre tiene solución. Unicamente cuando las señales de entrada son constantes a lo largo de cada intervalo de muestreo dicha solución existe. En efecto sea :

$$u(t) = u(k) = \text{cte. para } k\Delta t < t < (k+1)\,\Delta t \qquad (2.22)$$

La integración de (2.21) lleva a

$$x(t) = e^{At} x_o + \int_0^t e^{A(t-\tau)} B\, u(t)\, d\tau \qquad (2.23)$$

Si se impone como condición inicial la situación al comienzo del intervalo de muestreo k, es decir,

$$x(k) = x_o = x(k \cdot \Delta t) \qquad (2.24)$$

sabiendo que $u(\tau) = u(k) = cte$ puede obtenerse por sustitución directa en (2.23) el valor del vector de estados en el instante $(k+1)\cdot\Delta t$.

$$x(k+1) = e^{A\cdot\Delta t} x(k) + \left[\int_0^{\Delta t} e^{A(\Delta t - \tau)}\, d\tau \right] B\, u(k) \qquad (2.25)$$

Efectuando el cambio de variables

$$\eta = \Delta t - \tau \quad \text{y por tanto} \quad d\eta = -d\tau$$

se tiene

$$\int_{\tau=0}^{\tau=\Delta t} e^{A(\Delta t - \tau)}\, d\tau = \int_{\eta=0}^{\eta=\Delta t} e^{A\eta}\, d\eta \qquad (2.26)$$

Por simple inspección de (2.26), (2.25) y 2.20) se llega a :

$$P = e^{A \cdot \Delta t} \qquad (2.27)$$

$$Q = \left[\int_0^{\Delta t} e^{A\eta} \, d\eta \right] B \qquad (2.28)$$

que constituyen las ecuaciones de transformación del modelo discretizado al modelo continuo.

2.2.1.- Estudio de las ecuaciones de transformación.

Las ecuaciones (2.27) y (2.28) pueden escribirse en la forma,

$$A = \frac{1}{T} \ln (P) \qquad (2.29)$$

$$B = \left[\int_0^T e^{A\eta} \, d\eta \right]^{-1} Q \qquad (2.30)$$

si se define el logaritmo de la matriz cuadrada P como la función matricial $P \rightarrow A$ tal que siendo

$$e^{P} = A \quad \text{puede escribirse} \quad P = \ln A$$

La función $\ln P$ está siempre definida para cualquier P.

Si la matriz P está constituida por elementos reales, como es el caso tratado, la matriz A será real, si ningún autovalor de P es el real negativo.

La ecuación (2.30) admite una representación más sencilla si se tiene en cuenta que :

$$\int_0^T e^{A\eta} \, d\eta = A^{-1} \left[e^{AT} - 1 \right] \qquad (2.31)$$

Dado que $e^{AT} = P$ se obtiene sustituyendo

$$B = \left[P - I \right]^{-1} \cdot A \cdot Q \qquad (2.32)$$

donde $[P-I]$ es siempre no singular si A es no singular.

El único problema no trivial radica en la ecuación (2.29). Un primer método para el cálculo del algoritmo de una matriz cuadrada real se fundamenta en el desarrollo en serie :

$$l\,n(P) = \sum_{i=1}^{\infty} \frac{(-1)^{i-1}}{i} (P-I)^i \qquad (2.33)$$

Sin embargo se demuestra facilmente que para que el desarrollo converja es necesario que

$$\left| \lambda_i - 1 \right| < 1 \qquad \text{para todo} \quad i$$

donde λ_i son los autovalores de la matriz P.

Esta restricción es bastante severa y con evidente riesgo de no realizarse.

Otra forma de abordar el problema se expone en (ref.47) basándose en la expresión :

$$\ell_n(P) = \sum_{K=1}^{S} \left[\prod_{\substack{i=1 \\ i \neq K}}^{S} \left(\frac{P-\lambda_i I}{\lambda_k - \lambda_i} \right) \right] \ell_n(\lambda_K)$$

válida unicamente para el caso en que el polinomio mínimo de orden s tenga los autovalores distintos.

Acudiendo a la forma canónica de Jordan de (P-I) se muestra en (ref.47) otra forma de obtener el logaritmo de una matriz real. Aunque este método no presenta ninguna restricción en la estructura de la matriz P, sí presenta grandes dificultades de cálculo en cuanto que es necesario determinar los vectores propios de P.

Ante todo este tipo de inconvenientes se propone en el siguiente apartado una solución al problema de transformación planteado, eludiendo el cálculo del logaritmo de la matriz P.

2.2.2.- Algoritmo de transformación.

Obtenida una realización de $G(z)$ definida por el triplo $[P_c , Q_c, C]$, el problema de transformación se reduce, como se ha visto, a

encontrar una realización de G(s) definida por el triple [A,B,C] estando ligadas ambas realizaciones por el sistema de ecuaciones

$$P_c = e^{AT} \tag{2.35}$$

$$Q_c = (P_c - I)\, A^{-1}\, B \tag{2.36}$$

La matriz de transferencia buscada sería entonces :

$$G(s) = C\,(sI - A)^{-1}\, B \tag{2.37}$$

Supóngase una realización de G(s), definida por el triple $[A_c, B_c, C_c]$, que obviamente no satisfacerá las ecuaciones (2.35) y (2.36). Sin embargo, por similaridad puede escribirse :

$$\begin{aligned} A_c &= T_c\, A\, T_c^{-1} \\ B_c &= T_c\, B \\ C_c &= C\, T_c^{-1} \end{aligned} \tag{2.38}$$

donde T_c es una matriz no singular.

A partir de A_c y B_c por aplicación directa de (2.35) y (2.36), es posible determinar una segunda realización $[P, Q, C_c]$ de la matriz G(z). Esto es,

$$P = e^{A_c T} \tag{2.39}$$

$$Q = (P-I)\ A_c^{-1}\ B_c \tag{2.40}$$

Las dos realizaciones de la matriz $G(z)$ son similares via T_c. En efecto, sustituyendo (2.38) en (2.39) y (2.40) se llega a

$$P = e^{T_c (AT) T_c^{-1}} \tag{2.41}$$

$$Q = (P-I)\ T_c A^{-1}\ B \tag{2.42}$$

Desarrollando en serie, la exponencial matricial (2.41)

$$P = I + T_c A T_c^{-1}\ T + 1/2\ (T_c\ A T_c^{-1})^2\ T^2 + \ldots \tag{2.43}$$

y sacando factor común T_c por la izquierda y T_c^{-1} por la derecha se obtiene

$$P = T_c\ (I + AT + 1/2\ A^2 T^2 + \ldots)\ T_c^{-1} \tag{2.44}$$

donde el término entre paréntesis es precisamente el desarrollo en serie de e^{AT}. Teniendo en cuenta (2.35) se deduce que

$$P = T_c\ P_c\ T_c^{-1} \tag{2.45}$$

Por último, sustituyendo (2.45) en (2.42) y sacando Tc factor común por la izquierda se tiene :

$$Q = \left[T_c \, P_c \, T_c^{-1} - I \right] \quad T_c \, A^{-1} \quad B$$

$$= T_c \left[P_c - I \right] A^{-1} \quad B$$

donde la expresión que multiplica por la derecha a T_c es, en virtud de (2.36), la matriz Q_c. Se concluye por consiguiente que :

$$Q = T_c \quad Q_c \tag{2.47}$$

El problema de transformación, bajo el supuesto realizado, quedaría resuelto mediante la determinación de T_c ya que es trivial obtener el triple $[A,B,C]$ a partir de (2.38).

$$A = T_c^{-1} \quad A_c \quad T_c$$

$$B = T_c^{-1} \quad B_c \tag{2.48}$$

$$C = C_c \quad T_c$$

Tanto la suposición realizada, como la determinación de la matriz no singular T_c, constituyen dos problemas facilmente abordables si se acude a la siguiente realización canónica de $G(z)$ ó $G(s)$.

Sea el sistema lineal multivariable de p-entradas y m-salidas cuya descripción externa es la matriz de transferencia G(q) de dimensión (m x p)

$$T(q) = \frac{N(q)}{d(q)} \qquad (2.49)$$

donde :

d(q) es el mínimo común denominador mónico de los elementos de G(q) . Esto es,

$$d(q) = q^{n} + a_{n-1}\ q^{n-1} + \ldots\ a_{o} \qquad (2.50)$$

y

N(q) es la matriz polinomial (mxp) formada por los numeradores de los elementos de G(q) y que puede descomponerse en la forma

$$N(q) = N_{n-1}\ q^{n-1} + N_{n-2}\ q^{n-2} + \ldots\ No \qquad (2.51)$$

Una realización sin ceros acoplados con la entrada (ref. 49), es la definida por el triple[F, G, H] siguiente :

$$F = \begin{bmatrix} 0_p & I_p & 0 & \cdot & \cdot & \cdot \\ \cdot & 0_p & I_p & \cdot & \cdot & \cdot \\ \cdot & \cdot & 0_p & & & \\ \cdot & \cdot & \cdot\cdot & & & I_p \\ -a_o I_p & -a_1 I_p & \cdot & \cdot & & -a_{n-1} I_p \end{bmatrix} \qquad (2.52)$$

$$G = \begin{bmatrix} 0_p \\ \cdot \\ \cdot \\ \cdot \\ 0_p \\ I_p \end{bmatrix}$$

$$H = \begin{bmatrix} No & N_1 & \cdots\cdots & N_{n-1} \end{bmatrix}$$

donde F, G y H son matrices de dimensiones $(pn \times pn)$, $(pn \times p)$ $(m \times mn)$ respectivamente.

Con esta enumeración detallada se aprecia que el lenguaje de generación propuesto tiene potencia suficiente como para tratar con eficiencia todos los casos que puedan plantearse.

3.4. Justificación del lenguaje de generación .

Para la generación automática del analizador, en cuanto al código en el cual debe producirlo el sistema de programas, aparecen fundamentalmente las siguientes opciones :

a) Código de máquina de aquel computador sobre el que se desarrolle el sistema.

a') Código mnemotécnico o ensamblador del computador

b) Algún lenguaje de nivel medio o alto de los ya existentes.

c) Un lenguaje de programación nuevo, específico para esta aplicación.

Las opciones a y a' caen por su propio peso ya que tanto una como otra ligan el analizador generado a un sistema determinado o a una serie específica de computadores, anulando prácticamente la transportabilidad, lo que se traduciría en una utilidad mínima de los programas desarrollados .

La opción b , presenta grandes ventajas respecto a las anteriores puesto que la transportabilidad aumenta, ya que el analizador generado en un lenguaje evolucionado de los preexistentes puede

compilarse en cualquier máquina que disponga de un procesador para dicho lenguaje ; es este punto precisamente el que a la vez que una ventaja es un serio inconveniente, debido a que ni todos los equipos disponen de un compilador para un lenguaje dado, ni todos los compiladores de un mismo lenguaje tienen idénticas especificaciones, lo que conduciría a la necesidad de elegir un subconjunto de instrucciones del mismo suficientemente potente, y suficientemente generalizado de manera simultánea, como para poder escribir en él el analizador generado con garantías de un grado elevado de transportabilidad. Centrando la atención en algunos lenguajes, el FORTRAN es de los mas ampliamente difundidos y casi todos los computadores tanto de mediana y gran potencia, como algunos micro-procesadores avanzados (p. e., el PDP 11/03) disponen de compilador para el mismo, sin embargo presenta el gravísimo inconveniente de que el FORTRAN no soporta subprogramas recurrentes, por lo que sería preciso recurrir a algunos artificios en esamblador para conseguirlo, lo que reduce la alternativa al caso a' . Otro lenguaje de alto nivel que sí admite en su definición la escritura de rutinas recurrentes es el ALGOL , que presenta problemas en dos vertientes ; la primera de ellas estriba en que no es un lenguaje que se pueda decir"muy difundido" si se le compara por ejemplo con el FORTRAN, y la segunda consiste en que existen diferencias profundas en las implementaciones del ALGOL de unos sistemas a otros, lo que reduce la transportabilidad, llegando a afectar las diferencias citadas precisamente al aporte más esencial del ALGOL, la asignación dinámica de memoria y la posibilidad de definición de rutinas recurrentes, lo que anula toda su utilidad.

Por las razones anteriores se ha estimado conveniente la

elección de la alternativa c imponiendo como condición fundamental al lenguaje que contenga el menor número de instrucciones necesarias y que su sintaxis y sus especificaciones sean extremadamente simples, por lo que ha sido elegido como base el lenguaje precedentemente desarrollado por Griffiths (28)-(30) y que ha sido modificado aumentando su potencia y reduciendo su complejidad para alcanzar el lenguaje de generación propuesto.

Gracias a la simplicidad del lenguaje, un procesador para el mismo puede implementarse en cualquier máquina que disponga al menos de un ensamblador por rudimentario que sea, con poco esfuerzo y en escasos días ; disponiendo de un procesador de macroinstrucciones, el traductor puede obtenerse en media jornada por expansión de macros. En el contexto de la implementación realizada, se ha escrito un procesador de macros específico para este lenguaje que genera instrucciones de ensamblador del HP 21MX , y que por su modularidad, cambiando un subprograma de escritura, puede adaptarse a generar código ensamblador de otra máquina o bien otro lenguaje evolucionado ; como ejemplo puede citarse la adaptación al APL, preparada en un par de horas.

CAPITULO IV

ALGORITMOS DE VERIFICACION DE DETERMINISMO DESCENDENTE EN SRL

CAPITULO IV

ALGORITMOS DE VERIFICACION DE DETERMINISMO DESCENDENTE EN SRL

4.1. Descripción general del procedimiento de verificación de las condiciones de determinismo descendente.

El procedimiento de verificación implementado sigue en esencia las etapas generales de los algoritmos de Foster (33), (34) y de Griffiths y Peltier (18), (31) , aunque por las condiciones particulares de la gramática de la regla escrita en SRL, el modo de alcanzar los objetivos de cada una de estas etapas varía considerablemente.

Las etapas del procedimiento desarrollado son las siguientes .

a) Obtención de un vector que indica, para cada nombre de clase, si puede o no derivar sobre la cadena vacía (§ 4.2)

b) A partir del vector procedente, evaluando cada regla en particular puede determinarse si cumple la condición 4, tanto en la forma básica de Knuth para notación BNF como las extensiones a que da lugar en notación SRL (§ 4.3)

c) Obtención de las matrices binarias que representan las relaciones inicial, siguiente, final y vecindad.

Se han cambiado las distribuciones de las submatrices booleanas de las relaciones final y siguiente, con el objeto de que algunas submatrices no necesarias en las etapas finales del procedimiento, queden como objetos locales de la parte que los emplea, buscando un mejor aprovechamiento de la memoria disponible en el computador. Las matrices obtenidas se componen para determinar las relaciones representadas, en sentido estricto (§ 4.4)

d) Verificación de la condición 1, recurrencia por la izquierda (§ 4.5)

e) Verificación de la condición 3 para los nombres de clase (Condición 3 estricta de la gramática en notación BNF) (§ 4.6)

f) Reducción de la regla, eliminando elementos no necesarios en las verificaciones posteriores (§ 4.7.2)

g) Verificación de la condición 2 y de las extensiones de la condición 3 para SRL, así como generación de los conjuntos de símbolos terminales para elección de camino en los puntos de bifurcación (§ 4.7.3).

Los algoritmos desarrollados para cada una de las etapas se describen en los puntos siguientes.

Como nota histórica del desarrollo del trabajo, estos algoritmos se prepararon y analizaron inicialmente como un procesador basa

do en un autómata de pila (P.D.A.) con reducido número de estados , dado lugar a programas lentos, complejos y difíciles de depurar ; posteriormente se ensayó el análisis por descripción de la gramática de la frase que el algoritmo debía procesar, esto es, siguiendo la filosofía de funcionamiento del analizador que el propio sistema generaría , dando lugar a reducciones en programas del orden del 40 % , depuración más rápida y mayor facilidad cara a modificaciones posteriores. Se ha conservado el algoritmo de la etapa a en su forma original, en tanto que los demás han sido sustituidos por sus correspondientes versiones en descenso recurrente.

4.2. Algoritmo de obtención de los nombres de clase con posibilidad de derivar en la cadena vacía.

La base del algoritmo es la evaluación de cada una de las reglas en lógica ternaria (cierto - indeciso - falso) (35) , proceso que, como en el programa de Griffiths - Peltier se realiza reiteradamente , comenzando con la asignación a todos los nombres de clase del valor " indeciso ", y eliminando la regla cuando está decidido el valor del indicador de posible derivación en la cadena vacía (cierto o falso) ; el proceso iterativo termina cuando no quedan reglas indecisas o cuando no se resuelve ninguna indecisión tras una iteración completa.

Fácilmente se comprueba que la lógica ternaria es aplicable en este proceso, ya que las alternativas y las concatenaciones se tra-

ducen, respectivamente, por la suma y el producto en lógica ternaria; igualmente, puede comprobarse que, efectuando las representaciones que se indican seguidamente, las operaciones de suma y producto en lógica ternaria se reducen a la unión e intersección, bit a bit, de los valores binarios que los simbolizan internamente :

cierto : 11

indeciso: 01

falso : 00

Derivación en cadena vacía :

a) Aternativas : A | B

+	C	F	I	B
C	C	C	C	
F	C	F	I	
I	C	I	I	
A				

lógica ternaria

+	11	00	01	
11	11	11	11	
00	11	00	01	
01	11	01	01	
A				

Valor binario

b) Concatenación : A B

.	C	F	I	B
C	C	F	I	
F	F	F	F	
I	I	F	I	
A				

lógica ternaria

.	11	00	01	
11	11	00	01	
00	00	00	00	
01	01	00	01	

Valor binario

Para hacer más fácil la evaluación de las reglas, así como evitar la manipulación de información no necesaria, durante la primera iteración se realiza una reducción de las mismas, con los siguientes puntos :

- Se sustituyen los no terminales por el valor " falso ".
- Se eliminan las cadenas entre corchetes, sustituyéndolas por el valor " cierto ".
- Se reducen las estructuras de lista, dejando sólo la secuencia repetitiva, eliminando el separador y los meta-símbolos, ya que aquél no afecta al cálculo.
- Se eliminan las referencias a rutinas semánticas, irrelevantes en este proceso.

Tras la reducción anterior que, como ya se ha indicado, se efectúa sólo durante la primera iteración, se evalúan las reglas usando un mecanismo de pila, en la cual, junto con los valores ya especificados por la etapa precedente, así como los asociados a los nombres de clase (indecisos inicialmente), se guardan indicadores de posición de la regla resultante, que permiten eliminar símbolos no necesarios para la iteración siguiente, reducir alternativas (unión) , reducir cadenas (intersección), etc. , y evaluar la regla como una expresión en lógica ternaria segun el procedimiento ya indicado, obteniendo un resultado " cierto " , " falso " o " indeciso " ; para las dos primeras posibilidades, se anota el valor calculado y se elimina la regla del conjunto, en tanto que para la última se guarda la regla reducida equivalente, en la que solo aparecerán, junto con los meta-símbolos, nombres de clase no decididos cuando se procesó la regla ,y los valores " cierto " y " falso " .

Las operaciones que tienen lugar durante el proceso de cada uno de los elementos son :

1) Comienzo de la regla : iniciación del algoritmo.

2) Fin de la regla : reducción final y cierre de la cadena reducida generada.

3) Valor cierto ó falso : Si la cabeza de pila es separador de alternativas, paréntesis izquierdo o comienzo de regla, apilarlo, pero si es valor, multiplicar (intersección) por éste y sustituirlo por el resultado; si el resultado es " falso " ignorar resto de la alternativa, hasta hallar comienzo de otro paréntesis derecho o fin de regla.

4) Nombre de clase : tomar indicador de derivación en la cadena vacía del mismo ; si está indecisa, pasar nombre a la salida ; procesar como en 3.

5) Separador de alternativas : Si existía alternativa precedente, sumarlas (unión), reduciendo la regla generada si el resultado es cierto o falso ; si el resultado es " cierto ", ignorar todos los elementos hasta hallar un paréntesis derecho que cierre el nivel en proceso, o un fin de regla, si no , pasar separador a regla reducida y apilar copia.

6) Paréntesis izquierdo : Pasarlo a la regla reducida y apilar copia.

7) Paréntesis derecho : Cancelar posibles alternativas pendientes en la pila. Si el resultado es " indeciso ", pa

sar paréntesis derecho a la regla reducida, y si no, borrar regla reducida desde el final hasta el paréntesis izquierdo concordante. Desapilar valor y paréntesis izquierdo; procesar valor como en 3.

Como ya se ha expuesto, los demás metasímbolos no figuran por haber sido eliminados previamente.

4.3. Comprobación de la condición 4.

4.3.1. Descripción general.

En el desarrollo del trabajo, éste fué el primer algoritmo que se rediseñó aplicando la filosofía del descenso recurrente controlado por la gramática de la frase que debe procesar. De esta forma el algoritmo se reduce enormemente, tanto en su descripción como en el programa correspondiente.

La base de esta etapa del algoritmo es la evaluación de la regla de entrada, considerando que si un elemento de V puede derivar sobre la cadena vacía, toma el valor 1, y si no puede derivar, se le asigna el valor 0, lo que equivale a considerar sólo uno de los dígitos de la representación binaria del valor. De esta forma, la concatenación se traduce en el producto aritmético y la presencia de alternativas se procesa con sumas aritméticas de los valores equivalentes, de manera que, si existen dos o más alternativas nulas, la suma de los valores representativos de cada una de ellas será igual o superior a

dos, detectándose el error, exactamente cuando se sume el dígito asociado a una alternativa, al de las anteriores, restaurándose a 1 para continuar el proceso y detectar otras alternativas en la misma circunstancia .

Los casos adicionales planteados por la SRL se han resuelto como sigue :

a) Iteración $\alpha+$: verificar si el dígito de derivación en cadena vacía de α (ya calculado) es igual a 1, dando error.

b) Alternativas agrupadas por paréntesis $\beta_1(..|...|...)\beta_2$: Guardar valor del dígito asociado a la secuencia β_1 en una pila, evaluar y verificar contenido de paréntesis y multiplicar por valor apilado, decapitando la pila.

c) Alternativa/s opcional/es : simular la existencia de una alternativa nula, en adición al caso anterior, apilando un uno que la represente.

d) Estructura de lista : basta evaluar la secuencia base de la lista, sin considerar el separador de la misma, ya que es siempre un símbolo terminal.

4.3.2. Descripción del algoritmo

El algoritmo dispone a efectos de su realización práctica de los siguientes objetos locales, al nivel más alto del mismo :

- Una pila para guardar valores necesarios a conservar, pa-

ra suplir las deficiencias del FORTRAN IV en este aspecto (se ha desarrollado a tal efecto un grupo de subprogramas de gestión de pilas).

- LAST : variable que conserva el último valor procesado , necesario para la verificación de iteraciones.

- VS : Valor acumulado de la representación aritmética de la derivación en cadena vacía de la secuencia en proceso, considerada como se ha definido este concepto en la meta-gramática del punto 2.2.

- Tres contadores, para el número de paréntesis y corchetes izquierdos y comienzos de estructura de lista, para indicar al usuario la localización del error.

- Un indicador del número de orden de la alternativa en proceso, bien sea general o particular dentro del paréntesis o corchetes.

Con los objetos anteriores, éste algoritmo puede redactarse usando la descripción de la regla, con las acciones o rutinas semánticas incorporadas, tal como sigue :

```
REGLA -> "PR"  'INPILA'  DEFIN  "FR"

DEFIN -> 'INDEF' (. "," . "PS" SEC 'RDAL' .)

SEC -> 'INSEC' ( UNIDAD 'MULT' )+

UNIDAD -> "F",
```

```
"VT" 'FALSO' [ "+" ],
"VN" 'LOMVN' [ "+"  'ITER1' ],
'APIL'  ( "(" 'PI'  DEFIN  ")"  [ "+" 'ITER2' ],
          "[" 'CI'  DEFIN ( "]", "]+" ),
          "(." 'PL'  "VT" [ "F" ]+  "."  SEC ".)"
             )  'DESAP'
```

Las acciones indicadas son las siguientes :

- INPILA : Iniciar la pila; apilar indicador de principio de regla; iniciar a cero contadores; guardar - nombre de clase de la regla, para mensajes de error.
- INDEF : Iniciar a 1 el número de alternativa en proceso.
- AL : Apilar VS; sumar 1 a " número de alternativas", apilar indicador y número de la misma, para mensajes de error.
- RDAL : Si la cabeza de la pila es indicador de alternativa, decapitar dicha pila, recuperar el número de la alternativa, desapilar valor de la secuencia precedente, sumar a VS y si el resultado es mayor que 1, error 4.1 ó 4.2.
- INSEC : Poner VS a 1.
- MULT : Multiplicar VS por LAST y sustituir al primero

(VS * LAST --> VS).

- FALSO : Poner LAST a cero.

- LAMVN : Asignar a LAST el valor indicativo de derivación en cadena vacía del nombre procesado.

- ITER 1 : Si LAST vale 1 , error 4.3.

- ITER 2 : Si VS vale 1, error 4.4 (se ha preferido separar las iteraciones de símbolo del vocabulario y de estructuras entre paréntesis o corchetes para mayor claridad del mensaje de error cara al usuario).

- APIL : Apilar valor de VS.

- DESAP : Pasar valor de VS a LAST; decapitar la pila, desapilar valor anterior de VS.

- PI : Sumar 1 al número de paréntesis izquierdos procesados ; apilar indicador del paréntesis y número del mismo.

- CI : Suma 1 al número de corchetes izquierdos pasados; apilar indicador y número del corchete; apilar valor 1 e indicador de primera alternativa, simulando así la presencia de una alternativa ficticia con derivación en la cadena vacía.

- PL : Sumar 1 al número de estructuras de lista y apilar indicador.

Como se aprecia fácilmente, la mayor parte de la informa—

'ción que se almacena en la pila tiene por objeto situar el error lo mas exactamente posible, para facilitar al usuario la corrección del mismo.

4.3.3 Errores indicados.

E.4.1 : Dos o más alternativas nulas, con indicación de la situación mediante número de paréntesis izquierdo, de izquierda a derecha, y número de la alternativa.

E.4.2 : Iteración de una secuencia opcional con posible derivación interna en la cadena vacía.

E.4.3 : Iteración de nombre de clase con posible derivación vacía.

E.4.4 : Iteración de la estructura entre paréntesis, que tiene alguna derivación en la secuencia vacía.

4.4. Generación de tablas de símbolos iniciales y de siguientes.

4.4.1. Descripción general.

Este algoritmo tiene dos partes bien diferenciadas; la primera de ellas rellena las matrices booleanas de las relaciones binarias directas de iniciales, siguientes y finales, estando el proceso

controlado por la gramática de la regla, y la segunda efectúa las composiciones de las matrices y cierres transitivos de las relaciones que representan para obtener las matrices completas de iniciales y siguientes.

Se establecen tres matrices de trabajo , I , S y T, que contienen respectivamente :

- Matriz I : de tamaño N x (N + T), siendo N el número de símbolos del vocabulario no terminal y T el del vocabulario terminal. Representa las relaciones booleanas de iniciales directos.

- Matriz S : de igual tamaño que la anterior contiene

  a) zona N x N , la matriz booleana de la relación " final " , entre nombres de clase.

  b) zona N x T , la matriz booleana de la relación " siguientes " entre nombres de clase y terminales .

- Matriz T : matriz temporal, de tamaño N x (N + F) siendo F un cierto número de símbolos ficticios necesarios para la evaluación de relaciones en aquellos casos en que existan cadenas con posibilidad de iteración, estructura (.....) + ; el número F viene dado por la mayor profundidad que se presente en estructuras jerárquizadas de este tipo. La zona N x N contiene la relación binaria " siguientes " entre nombres de clase.

El método diseñado para llenar las matrices es el que se expone seguidamente . Dado que la notación SRL permite jerarquización

de definiciones, el método expuesto por Griffiths, de análisis de la regla de izquierda a derecha, tropieza con serias dificultades, precisamente al llegar a un punto en el que puedan surgir varias alternativas, ya que obligaría a seguirlas todas y cada una de ellas para cada símbolo que la preceda, si estos pueden derivar sobre la cadena vacía ; más concretamente, considérese una parte de una regla con la forma :

$$A \ \alpha_1 \ (\beta_1 | \beta_2 | \beta_3) \ \alpha_2$$

Si α_1 es una secuencia de V_N^+ que puede derivar en la cadena vacía , son siguientes de A todos los símbolos de α_1, así como los iniciales de β_1 , β_2 y β_3 ; si además algún β_i puede derivar en la cadena vacía también será necesario procesar α_2 para buscar los siguientes de A. A continuación se pasaría a considerar el primer símbolo de α_1 , repitiéndose el proceso anterior. Como se ve, los cálculos son similares al estudio de los siguientes de A, por lo que el algoritmo funcionando de ésta forma evolucionará lentamente.

El procedimiento que se ha elaborado, y programado, analiza la cadena precisamente en orden inverso, de derecha a izquierda, manteniendo una lista de los símbolos siguientes al elemento a procesar. Sea LS ésta lista de siguientes. Supongamos una situación en la que α_2 está formada por

$$\alpha_2 \equiv B \ \alpha_3$$

y que B es el símbolo siguiente a procesar. Todos los elementos de LS son siguientes de B , pudiendo incorporarse directamente a la matriz S o a la matriz T según sean terminales o nombres de clase. Seguidamente se pasa a considerar el no terminal B ; si este no puede deri-

var sobre la cadena vacía, B será inicial de α_2 , y por lo tanto siguiente directo de la parte pendiente de procesar a su izquierda, por lo que la lista LS se borra y se pone B en ella ; si B puede derivar sobre la cadena vacía, la lista de siguientes LS acumulada hasta B inclusive estará formada por los elementos de LS al procesar B, más el propio B. Al llegar a una bifurcación con varias alternativas entre paréntesis, bastará apilar una copia de la lista LS en curso, hallar las nuevas LS tras el proceso de β_3 , β_2 y β_1 , que se incorporarán a las matrices binarias T y S a medida que el análisis avanza , y al terminar cada alternativa β_2 y β_1 se efectúa una unión de las LS parciales de cada camino. Al llegar al paréntesis izquierdo, la LS calculada hasta α_2 inclusive ya no es necesaria, y la nueva LS , unión de las obtenidas tras el análisis de derecha a izquierda de β_1, β_2 y β_3 , será la lista de siguientes del último símbolo de la derecha de α_1 , progresando hacia la izquierda como anteriormente con α_2.

Este análisis de derecha a izquierda permite igualmente determinar los posibles iniciales de un nombre de clase, ya que al terminar de procesar toda la cadena en la que puede derivar, los siguientes acumulados hasta ese momento serán los iniciales del nombre de clase que se define ; más claramente, sobre un ejemplo :

A —> X α

Serán iniciales directos de A tanto X, como los siguientes directos de X en la regla que se está procesando, si X puede derivar sobre la cadena vacía, y estos precisamente son los acumulados en LS tras procesar α , que a su vez coinciden con los iniciales directos de la subcadena α , por lo que al llegar al símbolo de derivación (—>), en LS se tendrá la lista de iniciales de A en la regla, pudiéndose llevar directamente a la matriz I.

Por otra parte, si el proceso de cálculo inicia la lista LS con un indicador especial, unido al no terminal que se define en la regla, los nombres de clase unidos por la relación " final " de forma directa en la regla pueden obtenerse inmediatamente, como muestra el ejemplo siguiente :

Sea la regla $A \longrightarrow BCD$

y considérese que B, C y D pueden derivar sobre la cadena vacía ; sea A^F el indicador de final de A.

a) Situación inicial : $A \longrightarrow BCD$; $LS = \{A^F\}$

b) Proceso de D :

En LS sólo hay A^F , luego D es final de A y se anota en la matriz.

Como D deriva sobre la cadena vacía, se añade a LS.

c) Proceso de C : $A \longrightarrow BC$ —— ; $LS = \{A^F , D\}$

De LS se deduce : C es final de A, D es siguiente de C y se anota.

Como C deriva sobre la cadena vacía, se añade a LS.

d) Proceso de B : $A \longrightarrow B$ —— ; $LS = \{A^F, D, C\}$

De LS, B es final de A; D y C son siguientes de B. Se añade B a LS por tener una derivación nula.

e) Final $A \longrightarrow$ —— $LS = \{A^F, D, C, B\}$

La presencia de A^F en LS solo comprueba que A tiene alguna derivación nula. B, C y D son iniciales directos de A y se anotan.

Este ejemplo tan sencillo muestra la potencia del método de-

sarrollado, que procesa la cadena sin vuelta atrás, determinando conjuntamente las relaciones de iniciales, siguientes y finales.

El método presenta dos puntos más complicados, exponiéndose seguidamente la solución adoptada.

a) Iteraciones.

a1) Iteración simple de nombre de clase. El nombre de clase es : siguiente de sí mismo. Si no deriva sobre la la cadena vacía, se limpia LS, y se coloca en ella dicho nombre de clase.

a2) Iteración de estructura con alternativas :$(\beta_1 \mid \beta_2) + \alpha$ Son siguientes de β_1 y β_2 no sólo los iniciales de α sino también los iniciales de β_1 y β_2 .
El problema se ha resuelto generando un símbolo ficticio F_x que representa al conjunto de iniciales de β_1 y β_2 , conjunto que quedará unívocamente determinado al procesar el paréntesis izquierdo, y podrá sustituirse en las matrices binarias ; si dicho símbolo ficticio no desaparece al terminar de procesar la iteración, será señal de que existe una posible derivación nula, que está prohibida en la iteración , y señalada ya como error, al verificar la cuarta condición de determinismo descendente, por lo que aquí no se genera ningún mensaje adicional.

b) Estructura de lista. Por la representación interna de la regla, es posible anticiparse al proceso del símbolo separador de la lista , e incorporarlo a LS . Si se quisiera un algoritmo estrictamente -

descendente, avanzando paso a paso, o bien se quisieran ampliar - las posibilidades del separador, podría recurrirse a un artificio como el aplicado en la iteración de estructura con alternativas.

Una vez calculadas las matrices binarias de las relaciones directas, basta efectuar los pasos siguientes, iguales a los presentados por Griffiths :

a) Cierre transitivo de la relación binaria " inicial " - (matriz I).

b) Sustitución en la matriz S de la relación binaria " siguiente " entre nombres de clase y los iniciales correspondientes (composición de T e I, uniéndola a S).

c) Cierre transitivo de la relación " final " , arrastrando los siguientes (matriz S), obteniendo así la matriz de la relación binaria " vecindad ".

4.4.2. Algoritmo de llenado de tablas.

a) Variables del algoritmo :

- LS : Lista de siguientes acumuladas para la secuencia en proceso, hasta el final de la regla.

- LP : Lista de siguientes acumulados hasta el comienzo del nivel de jerarquización en proceso.

- NF : Contador de símbolos ficticios generados, para - estructuras jerarquizadas con iteraciones.

- Una pila para guardar resultados parciales.

b) Gramática de la regla invertida, con acciones incluidas.

```
REGLA -> "FR"  'INPR'  DEFIN  "PR"  'TBIN'

DEFIN -> SEC  "PS" [ 'GLS'  ( "," 'RSLP'  SEC  "PS"  'SUM' )+  'RSL' ]

SEC -> UNIDAD+

UNIDAD -> "F",
          "+"  ( "VT" 'PVT',
                 "VN" 'PVN'  'PVNM',
                 ")"  'GF' 'NL'  DEFIN 'FL' 'TF' "(" ),
          "VT" 'PVT',
          "VN" 'PVN',
          ")"  'NL'  DEFIN  "(" 'FL',
          "]"  'NL'  DEFIN  "[" 'CI' 'FL',
          ".)" 'PEL'  SEC  "."  [ "F" ]+  "VT"  "(."
```

c) Funciones semánticas.

- INPR : Iniciar pila; vaciar LP ; iniciar LS con indicador de relación final con el nombre de clase - cuya regla se analiza; iniciar a cero NF.
- TBIN : Poner a 1 los puntos necesarios en la tabla de iniciales, según contenido de LS.
- GLS : Guardar copia de LS en la pila.

- RSLP : Copiar LP en LS (al salir al nivel exterior en una jerarquía de paréntesis).

- SUM : Unir LS a la lista en la cabecera de la pila , dejando el resultado en ella (alternativas).

- RLS : Limpiar LS, y llenarlo con la lista en la cabecera de la pila, decapitándola.

- PVT : Limpiar LS , poner inidicador del terminal en LS.

- PVN : Anotar LS en siguientes a nombre de clase ; si dicho nombre no tiene ninguna posible derivación en la cadena vacía , limpiar LS ; añadir indicador del nombre de clase a LS.

- PVNM : Anotar nombre de clase como siguiente de sí mismo.

- GF : Generar símbolo ficticio : sumar 1 a NF, añadir indicador del ficticio a LS.

- NL : Apilar LP ; copiar LS en LP .

- FL : Desapilar lista de cabeza de pila y pasarla a LP destruyendo su contenido anterior.

- TF : Sustituir indicador del símbolo ficticio por el contenido de LS ; destruir ficticio generado y restar 1 a NF.

- CI : Añadir LP a LS (alternativa opcional).

- PEL : Tomar separador de la lista y añadirlo a LS.

4.5. Verificación de la condición 1.

Es inmediata a partir de la matriz I_+ generada en el punto anterior. Basta inspeccionar la diagonal principal de la submatriz $N \times N$, y si existe algún 1 en ella, el símbolo correspondiente es un nombre de clase recurrente por la izquierda.

4.6. Verificación de la condición 3 para nombres de clase.

Puede verificarse hallando la intersección de los dos conjuntos de símbolos terminales, los iniciales y los siguientes, de los nombres de clase que tengan alguna derivación nula, informaciones suministradas en el vector y las tablas calculadas anteriormente en 4.2 y 4.4.

4.7. Verificación de la condición 2 y extensiones de la 3, con generación de listas de decisión.

4.7.1. Descripción general.

Básicamente, el procedimiento es el mismo que para la determinación de las matrices binarias, pero en este caso, cuando se procesa un nombre de clase, lo que se añade a la lista LS son los símbolos terminales iniciales, directos e indirectos , de dicho nombre de clase, verificando la presencia de símbolos comunes, lo que comprueba la condición 3 y sus extensiones, y efectuando la misma comprobación al unir cadenas de iniciales de distintas alternativas (verificación de

la condición 2).

Para procurar que el volumen de información no necesaria que se procese sea mínimo, se efectúa previamente un proceso de reducción - de la regla, eliminando los símbolos irrelevantes (elementos en secuencia tras un terminal o un nombre de clase sin posible derivación nula) para la determinación de las cadenas de iniciales en cada punto en que sea necesario tomar una decisión (alternatives, iteraciones, cadenas - opcionales , etc.).

Ambos algoritmos, tanto el de reducción de la regla como el de verificación, están diseñados por análisis descendente controlado por la gramática de la regla.

4.7.2. Algoritmo de reducción.

a) Variables principales.

- Contadores de número de paréntesis, corchetes, comienzo de listas y alternativas.
- Valor de posibilidad de derivación en cadena nula de la secuencia en proceso , identificación del último símbolo del vocabulario que se haya procesado.

b) Gramática de la regla, con acciones incorporadas.

```
REGLA -> "PR"  'INPR'  DEFIN  "FR"  'FNPR'

DEFIN -> 'INDEF' "PS"  SEC  [ "," 'PALT'  "PS" SEC 'SALT' ]+
```

```
SEC ->   'INSEC'  UNIDAD+  'TSL'

UNIDAD -> "F",
          "VN" 'GVN'  ( "+" 'TSLC'  'VNM',  'PVN' ),
          "VT" 'GVT'  ( "+" 'TSLC'  'VTM',  'PVT' ),
          'TSLC'  ( "(" 'CPI'  DEFIN ")" 'CPD'  [ "+" 'PMAS' ],
                    "[" 'CCI'  DEFIN  ( "]" , "]+" )  'CCDM',
                    "(." 'PCL'  "VT"  'SLIS'  [ "F" ]+  "."  SEC ".)"

                        'PFL'  )
```

c) Funciones semánticas.

- INPR : Iniciar pila, regla reducida de salida y contadores.
- FNPR : Terminar regla reducida, añadiendo control de fin de regla.
- INDEF: Poner a cero el número de la alternativa en proceso.
- PALT : Apilar el valor del indicador de derivación en cadena vacía (DCV) ; sumar 1 a número de alternativa, pasar copia a la regla reducida de salida.
- SALT : Desapilar indicador de alternativa y recuperar número de la misma; desapilar valor DCV anterior y hallar la suma booleana con DCV en curso, sus-

tituyendo a este último.

- INSET : Poner DCV a 1.

- TSL : Cerrar subcadena en regla de salida (indicador de fin de subcadena, número de elementos en la misma y valor parcial de DCV).

- GVN : Guardar identificador del no terminal.

- TSLC : Si la subcadena última en la regla reducida no está vacía, cerrarla (TSL).

- VNM : Pasar no terminal y símbolo "+" a la regla reducida. Asignar a DCV el indicador de posible derivación en cadena vacía del no terminal procesado.

- PVN : Si DCV es 1, pasar no terminal a cadena reducida, asignar a DCV el indicador de derivación en la cadena vacía del no terminal. Si DCV vale 0, ignorar no terminal.

- GVT : Guardar identificador del terminal.

- VTM : Pasar a regla reducida el terminal y el símbolo "+" ; poner DCV a cero.

- PVT : Si DCV vale 1, pasar terminal a cedena reducida y poner DCV a cero ; en caso contrario, ignorar terminal.

- CPI : Sumar 1 a contador de paréntesis izquierdos;pasar paréntesis y contador a regla reducida;apilar copia del mismo.

- CPD : Desapilar indicador del paréntesis izquierdo y

recuperar número del mismo ; pasar a la salida el paréntesis derecho con el mismo número.

- CCI : Igual que CPI, pero con corchete izquierdo.
- CCDM : Desapilar indicador del corchete izquierdo y recuperar número del mismo ; pasar a la salida el último metasímbolo procesado (∃ ó ∄ +) junto con el número de identificación; poner DCV a 1.
- PCL : Igual que CPI, pero para indicador de comienzo de lista.
- SLIS : Apilar separador de lista.
- PFL : Pasar "." a la salida; desapilar separador de lista y añadirlo a la regla reducida; desapilar indicador de principio de lista; recuperar número del indicador y pasar cierre de lista a la salida con dicho número.

4.7.3. Algoritmo de verificación.

a) Variables empleadas :

- LS : Lista de iniciales de la secuencia en proceso.
- LP : Id. para secuencia del nivel inmediatamente superior.
- LD : Cadena de elementos del conjunto de iniciales, respecto a los que hay que decidir la alternativa a tomar, valor de salida del algoritmo.

- LI : Lista de iniciales de un nombre de clase o de un conjunto de ellos, junto con los terminales de la misma subcadena.

- NALT: Número de la alternativa en proceso, para mensajes de error localizados.

- Una pila para almacenamiento temporal.

b) Gramática de la regla reducida reflejada, con funciones semánticas incluídas.

```
REGLA -> "FR"  'PFR'  DEFIN  "PR"  'PPR'

DEFIN -> SEC  [ 'LSLD'  ( 'GLS'  ","  SEC 'LSLD' 'SUM' )+  ]

SEC -> UNIDAD+

UNIDAD -> "SLIS"  'RTLI',
          "+"  ( "VT"  'PVTM',
                 "VN"  'PVNM',
                 'APIL'  'INITER'  ")"  DEFIN  'LSLD'  'FNITER'
                       'DPIL'  "("  ),
          'APIL'  ( ")"  DEFIN  'DPIL'  "(" ,
                    ( "]" , "]+" )  DEFIN  'LSLD' 'CI'  'DPIL'  "["  ),
          ".)"  "VT"  'VSL'  "."  SEC  "(."
```

c) Funciones semánticas.

- PFR : Iniciar la pila; almacenar indicador de fin de regla; iniciar LS y LP con el conjunto de siguien-

tes del nombre de clase cuya regla se analiza.

- PPR : Cerrar conjunto de listas de decisión.

- LSLD: Pasar LS a la cadena de salida.

- GLS : Apilar indicador de alternativa.

- SUM : Desapilar indicador de alternativa y guardar número de la misma ; desapilar lista de iniciales, hallar unión e intersección con LS. Si la intersección no está vacía, error condición 2.

- RTLI: Construir lista de iniciales de subcadena; si el indicador de derivación en cadena vacía de la sublista vale cero, limpiar LS ; hallar unión de ambas listas de iniciales.

- PVTM: Si el terminal está en la lista LS, error. Limpiar LS y poner sólo el terminal.

- PVNM: Hallar iniciales de nombre de clase, tomándolos de la matriz I; hallar intersección con LS, y si no es vacía, error ; pasar iniciales del nombre de clase a listas de decisión.

- APIL: Apilar LP ; copiar LS en LP; apilar último metasímbolo.

- DPIL: Desapilar metasímbolo; desapilar lista y guardar en LP.

- INITER: Pasar indicador de iteración a LD; apilar copia del indicador.

- FNITER: Desapilar indicador de iteración; enlazar indicador con situación de la lista de decisión.

Si la intersección de LS con LP es no vacía , error.

- CI : Hallar unión e intersección de LS y LP ; si la intersección no está vacía, error.

- VSL : Si el separador está en LS, error.

d) Errores indicados :

E 2. 0 - Alternativas con iniciales no disjuntos.

E 3. 2 - Iteración de símbolo terminal no disjunto de los siguientes.

E 3. 3 - Iteración de nombre de clase con iniciales no disjuntos de los siguientes.

E 3. 4 - Iteración de estructura entre paréntesis con iniciales no disjuntos de los siguientes.

E 3. 5 - Cadena opcional con iniciales no disjuntos de los siguientes.

E 3. 6 - Separador de lista no disjunto de los posibles siguientes a la misma.

En los mensajes se indica la situación del error así como los símbolos terminales comunes.

CAPITULO V

IMPLEMENTACION EN UN COMPUTADOR DE TAMAÑO MEDIO

CAPITULO V

IMPLEMENTACION EN UN COMPUTADOR DE TAMAÑO MEDIO

5.1. Visión general del sistema.

5.1.1. Configuración del equipo.

La implementación del generador de analizadores se ha efectuado sobre un sistema de Hewlett Packard basado en un minicomputador HP 21MX ; constituido por :

- Procesador HP 2112A
- Memoria central de 72 K palabras de 16 bits.
- Sistemas de protección de memoria, recuperación de fallo de tensión y direccionamiento de páginas por mapas dinámicos (DMS).
- Unidad de disco HP 7905 A , con una capacidad de 15 Mb.
- Lectora de tarjetas HP 2892A , de 600 t.p.m.
- Lectora de cinta HP 2748B , de 300 c.p.m.
- Impresora de líneas HP 2706A, de 200 l.p.m.
- Unidad de pantalla alfanumérica HP 2640 A , que actúa como consola de operador y como terminal.

- Teletipo marca TELETYPE, modelo 33, que funciona como - terminal y como perforador de cinta.

El sistema operativo actual es el RTE-III , el cual, por - la estructura del DMS disponible durante el desarrollo de este trabajo, divide la memoria en particiones, de manera que el espacio ocupado por la misma, más el ocupado por el ejecutivo no puede exceder de 32K, de modo que la máxima partición definible en este sistema es de 14 K palabras (Nota : Hewlett-Packard ha desarrollado un nuevo sistema , RTE-IV , sobre la base de un nuevo DMS que permite direccionar particiones hasta 1.024 K, que está en el mercado desde el 1º de Mayo del presente año).

El RTE-III actual dispone de los procesadores para FORTRAN IV , una versión no estándar del ALGOL , llamada HP-ALGOL, ensamblador sin procesador de macros, y BASIC conversacional multi-usuario.

5.1.2. Programas desarrollados .

Dadas las características anteriores, y como el HP ALGOL no soporta asignación dinámica de memoria ni rutinas recurrentes, se ha elegido el FORTRAN IV como lenguaje principal para la escritura - de los programas, ya que la versión disponible es bastante potente y su compilador produce código más depurado que el HP-ALGOL. En las aplicaciones concretas en que era imprescindible, se han escrito programas en ensamblador del sistema para suplir las deficiencias del - FORTRAN IV .

Los programas desarrollados han sido :

- MCMP : Lectura y verificación de sintaxis de las reglas, con generación de código interno.

- PCLL1: Verificación de las condiciones de determinismo y generación de las listas de decisión.

- MCGEN: Producción del analizador en el lenguaje de generación.

- PMMC : Expansión de las instrucciones del lenguaje de generación, produciendo las instrucciones de ensamblador necesarias.

Igualmente, se han desarrollado diversas bibliotecas de subprogramas, algunas de las cuales pueden ser de utilidad general, aunque inicialmente se diseñaran para este problema.

5.2. Forma interna de las reglas de la gramática.

Las reglas de la gramática se representan internamente mediante una cadena de datos, organizados linealmente, que tiene como eslabón básico una palabra de 16 bits con la estructuración siguiente :

- bit 15 (bit S) : toma el valor 0 si ningún otro eslabón de la cadena enlaza con este, y 1 en caso contrario. Tiene como misión fundamental señalar los puntos en que hay que generar una etiqueta de referencia.

- bits 10-14 (bits T) : indican el tipo de elemento que representa el eslabón. Los tipos básicos establecidos son los siguientes :

  1 - Comienzo de regla.

  2 - Fin de regla.

  3 - Comienzo de secuencia.

  4 - Separador de alternativas (',').

  5 - Paréntesis izquierdo.

  6 - Paréntesis derecho.

  7 - Corchete izquierdo.

  8 - Corchete derecho.

  9 - Iteración simple ('+').

  10 - Iteración opcional ('] +').

  11 - Comienzo de estructura de lista ('(.').

  12 - Punto central de la lista ('.').

  13 - Fin de lista ('.)').

  14 - Terminal .

  15 - No terminal.

  16 - Acción, rutina semántica .

No se ha establecido símbolo especial para la estructura de lista opcional ('[._ _ . _ _ _ .]') ya que es equivalente exactamente a la estructura normal, entre corchetes ('[(. _ _ . _ _ _ .)]'),

tanto en el proceso de los algoritmos como en el de código generado.

No se usa el cero, y algunos de los tipos restantes, hasta 31, se emplean en etapas del proceso de verificación de las condiciones de determinismo y generación de listas de decisión.

- bits 0 a 9 (bits E) : el contenido de estos bits varía de unos tipos a otros, siendo un número de elemento del vocabulario V_N ó V_T ó del conjunto de acciones, o una dirección de enlace a otro eslabón de la cadena. Los contenidos en cada caso son:

  Tipo: 1 - Nombre de clase que se define en la regla.

  2 - Cero.

  3 - Enlace al comienzo de la secuencia alternativa siguiente, o cero si es la última.

  4 - Enlace al final de todas las alternativas de su mismo nivel, esto es, el comienzo de la continuación de la cadena al nivel precedente.

  5 - Enlace al eslabón siguiente al paréntesis derecho concordante.

  6 - Enlace al paréntesis izquierdo concordante.

  7 - Igual que 5, para el corchete derecho.

  8 - Igual que 6, para el corchete izquierdo.

  9 - Enlace con el elemento precedente si la iteración se refiere al terminal o no terminal., o al paréntesis izquierdo si se refiere a una estructura entre paréntesis.

  10 - Igual que 8.

  11 - Enlace al siguiente punto central (tipo 12).

12 - Enlace con eslabón de comienzo de lista (tipo 11).

13 - Enlace al separador de la lista.

14 - Número representativo del símbolo terminal.

15 - Id. del símbolo no terminal.

16 - Id. del nombre de la función.

Con este tipo de estructura, la forma interna de la regla permite generar directamente las instrucciones del analizador, al incorporarle los conjuntos de decisión , y además el espacio que ocupa en memoria es menor que el que sería necesario usando estructuras clásicas de listas .

5.3. Análisis sintáctico de las reglas y producción de la forma interna de la misma.

El algoritmo que verifica la corrección de la sintaxis, y genera la forma interna de las reglas,es un caso típico de analizador, por lo que puede ser descrito gramaticalmente, con las funciones incorporadas, tal como se indica seguidamente.

```
REGLA -> "NOTERMINAL" 'PR' "->" DEFINICION 'FR' ";"

DEFINICION -> (. "," 'AL' . 'PS' UNIDAD+ .)

UNIDAD -> "TERMINAL" 'T' [ "+" 'BUCL' ],
          "NOTERMINAL" 'N' [ "+" 'BUCL' ],
          "FUNCION" 'F',
```

"(" ('PI' DEFINICION ")" 'PD' ["*" 'BUCLE'],

LISTA ")"),

"[" 'CI' (DEFINICION "]" 'CD' ["*" 'AST'],

LISTA "]" 'CD')

LISTA -> "." 'PL' "TERMINAL" 'T' ["FUNCION" 'F']+ "." 'ML'

UNIDAD+ "." 'FL'

Las acciones realizadas por cada una de las funciones semánticas son las siguientes.

- PR : Generar bloque de comienzo de regla, con nombre de clase que se define.
- FR : Generar fin de regla; cerrar alternativas del nivel más exterior si las hay.
- AL : Generar separador de alternativas; enlazar comienzo de secuencia precedente con siguiente eslabón.
- PS : Generar bloque de comienzo de secuencia.
- T : Generar bloque tipo terminal.
- N : Id., tipo no terminal .
- F : Id., tipo función semántica.
- PI : Id., tipo paréntesis izquierdo.
- PD : Id., tipo paréntesis derecho; enlazar con paréntesis izquierdo concordante; enlazar finales de alternativas entre ambos con eslabón siguiente y marcarlo como referido.

- BUCL : Generar bloque de iteración con enlace al precedente, que se marca como referido.

- BUCLR: Generar bloque de iteración, con enlace al paréntesis que abre la definición iterativa.

- CI : Generar bloque de corchete izquierdo.

- CD : Generar bloque de corchete derecho; enlazar con corchete izquierdo concordante; enlazar éste con el siguiente al corchete derecho y marcarlo como referido.

- AST : Transformar último eslabón en tipo iteración opcional (ε10).

- PL : Generar eslabón 'principio de lista'.

- ML : Generar eslabón 'punto medio de lista'; enlazar principio de lista con eslabón siguiente y marcarlo como referido.

- FL : Generar bloque de fin de lista; enlazar con eslabón siguiente al comienzo lista precedente y marcarlo como referido.

El algoritmo precedente prueba, una vez más, la potencia del método elegido, y que las diversas etapas de análisis del generador de analizadores presentado pueden obtenerse automáticamente usando el propio generador de analizadores.

---- ----

5.4. Programa MCMP.

El analizador de la sintaxis de la gramática de entrada está desarrollado sobre la base del algoritmo anterior, el cual se implementa como un subprograma, al que se le añade otro subprograma adicional que efectúa el análisis lexicográfico, sobre la base de un sencillo autómata finito, y otro que lleva la cuenta de los no terminales cuya regla ha sido definida, para determinar aquellos que se intenten definir más de una vez y aquellos otros que no hayan sido definidos. En caso de que no existan errores de sintaxis, se pasa al módulo siguiente, el programa PCLL1, que verifica las condiciones de determinismo.

Respecto a la tabla de símbolos generada por este módulo, se ha adoptado un sencillo método de organización lineal, esto es, tabla no clasificada, que si bien no es un método óptimo, resuelve el problema de modo temporal, ya que el objeto del trabajo es el desarrollo y prueba de algoritmos de verificación de determinismo, según las condiciones LL(1) de Knuth, sobre la base de la notación SRL ; no obstante, y puesto que el acceso a la tabla de símbolos lo realiza un solo subprograma, cambiando éste puede adoptarse cualquiera de los métodos de clasificación de tablas de símbolos, como pudiera ser una organización por residuos.

Se ha incluido en este programa la posibilidad de definir un vocabulario inicial, que luego puede ser ampliado durante el proceso de las reglas. Esta opción permite que el usuario seleccione de — cierta forma el valor decimal que se asigna a los identificadores de terminales, no terminales y funciones, siendo particularmente intere-

sante conocer exactamente esta asignación para los terminales, de cara a la escritura del analizador lexicográfico, que proporciona el siguiente símbolo a procesar ; igualmente, la opción es interesante, puesto que si el lenguaje de escritura de las funciones semánticas no permite la definición de varios puntos de entrada en un subprograma (caso del FORTRAN IV del HP 21 MX disponible), un pequeño cambio en el generador producirá llamadas a una única rutina del usuario, y en la que un argumento determinará la función a realizar.

El subprograma que constituye el analizador ha sido escrito en el lenguaje de generación propuesto, ya que el procesador de macros del mismo, PMMC, fué escrito y puesto a punto con anterioridad, de forma similar a las técnicas de programación de compiladores por " boot strapping ".

Se ha incorporado igualmente el análisis de una tarjeta de control con opciones de listado, generación , traza del proceso de análisis , y selección del modo de funcionamiento del analizador generado, pudiendo actualmente elegirse entre programa principal y subprograma compatible con las llamadas estándar del sistema.

El resultado final de todo el programa es un fichero, que contiene las reglas de la gramática en forma interna, así como la tabla de símbolos de la misma.

5.5. Programa PCLL1.

Este programa toma el fichero generado por MCMP y, con a-

yuda de dos ficheros auxiliares, verifica las condiciones de determi-- nismo descendente LL(1), generando además los conjuntos de terminales necesarios para la elección de alternativas, conjuntos que se obtie--- nen como resultado en uno de los ficheros de trabajo anteriores. Si -- no hay errores, enlaza la ejecución del programa MCGEN.

Dada la reducida memoria disponible, el programa está dividido en segmentos que se solapan en el transcurso del análisis, tal como se marcan en los apartados siguientes ; dado que el sistema RTE-III no tiene ninguna forma estándar que permita llamar a segmentos en " overlay " y recuperar posteriormente el control en el programa principal si este está escrito en FORTRAN IV , ha sido preciso desarro--- llar subprogramas en lenguaje ensamblador a tal efecto.

5.5.1. Modulo MCMP2.

Determina el vector de derivación en la cadena vacía de la etapa a del algoritmo (§4.2); si dicho cálculo lleva al procesador a un bucle con símbolos indeterminados, se escribe un mensaje de - error, junto con los nombres de clase implicados y se les asigna el - valor " cierto ", para seguir avanzando aunque no se genere código final, lo que permite detectar errores posteriores.

Un subprograma adicional, según el algoritmo descrito en 4.3., verifica la condición 4 para todas las reglas, señalando los - errores que se presenten.

--- ---

5.5.2. Módulo MCMP3.

Este segmento realiza las etapas c , d y e del algoritmo expuesto en 4.1, correspondiente a los apartados 4.4,4.5 y 4.6 . Las matrices booleanas se manejan empaquetadas (a bit por punto de las mismas), habiéndose desarrollado subprogramas de gestión de las tablas,que permiten poner a 1 un punto, leer un valor concreto, hallar uniones de filas y haller el cierre transitivo de las relaciones binarias que representan, entre otras funciones, que se describen más detalladamente en 5.8.3.

5.5.3. Módulo MCMP4.

Realiza las etapas f y g del algoritmo de 4.1, según el método expuesto en 4.7.2 y 4.7.3. El resultado es, para cada regla , una nueva cadena de conjuntos que contienen los símbolos terminales que permiten escoger la alternativa adecuada (o señalar el error) , colocados de manera que resulten concordantes con la regla primitiva, para mayor facilidad en la generación de código; estas nuevas cadenas se almacenan en un fichero auxiliar.

5.6. Programa MCGEN.

Este programa produce el analizador escrito en el lenguaje de generación de 3.1 . Aunque la generación de código es inmediata partiendo de los ficheros precedentes, el de reglas en forma interna y el de conjuntos de listas de decisión, se efectúa un proceso previo en la regla tendente a mejorar el código resultante, reduciendo y

eliminando las transferencias encadenadas y las etiquetas no necesarias ; igualmente, en esta etapa se determinan los códigos de error a emitir como consecuencia de la aparición de un caracter no esperado, que no concuerde con ninguna de las alternativas posibles en este momento.

El analizador generado puede obtenerse en un fichero, en cinta perforada, o sencillamente en papel impreso.

5.7. Programa PMMC.

La misión fundamental de este programa es suplir la deficiencia del ensamblador el HP 21MX, que no dispone de procesador de macros, habiendo sido desarrollado como un expansor específico para las instrucciones del lenguaje de generación, tarea más simple y ejecución más rápida que la realización y puesta a punto de un procesador generalizado de macros.

No obstante ser específico del HP 21MX, su concepción, igualmente controlada por la gramática, es muy modular, y cambiando un subprograma que solo contiene órdenes de escritura, puede transformarse, generando código para cualquier otro ensamblador, e incluso para lenguajes de alto nivel. Igualmente, en vez de generar código para un lenguaje, puede transformarse el analizador en una serie de tablas que sean procesadas posteriormente por un interpretador o metacompilador.

Este procesador de macros específico tiene la estructura que se indi

ca a continuación

a) Gramática.

```
PROGRAMA -> 'FØ' "COMIENZO" 'F1' LINEA+ "FIN" 'F3'

LINEA -> [ "NOMBRE" 'F4' ] INSTRUCCION

INSTRUCCION -> ( "GOTO" 'F5', "GOSUB" 'F6', "CALL" 'F11',
                 "DECIDE" 'F9' LISTA "," "CONDICION" 'F14',
                 "ROUTINE" 'F17') "NOMBRE" 'F12',
               "RETURN" 'F7',
               "CHECK" 'F8' "CARACTER" 'F13',
               "FAULT" 'F10' ( "CARACTER" , "NUMERO" ) 'F15'

LISTA -> "CARACTER" 'F16',
         "(" (. "," . "CARACTER" 'F16' .) ")"
```

b) Funciones

- FØ : Iniciación.
- F1 : Análisis de tarjeta de control.
- F2 : Generación del código final de la línea.
- F3 : Generar línea de fin de programa.
- F4 : Guardar " nombre " en N1.
- F5 a F11 y F17 : Guardar tipo de instrucción.

- F12 : Guardar " nombre " en N2.

- F13 : Guardar carácter.

- F14 : Guardar condición.

- F15 : Guardar dato en " código de error ".

- F16 : Guardar carácter en lista de decisión.

Como ya se ha indicado, es cuestión de cambiar F2 para -- adaptar el programa a la producción de instrucciones en otro lenguaje distinto del ensamblador del HP 21MX.

5.8. Bibliotecas de subprogramas.

Se han desarrollado seis bibliotecas de subprogramas, una de las cuales permite implementar las diversas instrucciones del analizador con facilidad, y las cinco restantes son de aplicación general, aunque inicialmente se implementaron para su utilización por el - generador de analizadores. Estas bibliotecas se presentan en los apartados siguientes.

5.8.1. Subprogramas de base del analizador generado.

Consta en esencia de cinco subprogramas cuyas misiones específicas son :

- GOSUB : Implementación de la llamada a la rutina generada para una regla, apilando punto de llamada, pa

ra permitir definiciones recurrentes. Si el espacio disponible para la pila está agotado, se genera un mensaje de error y no se efectúa la llamada, pudiendo elegir el programador entre continuar a pesar del error o abortar la ejecución. Un punto de entrada adicional, GSB, hace compatible este subprograma con las llamadas normales del FORTRAN IV y del ALGOL del sistema, con vista a futuras aplicaciones.

- RETUR : Salida de una rutina recurrente, con devolución del control al punto de la llamada precedente. Si la pila está vacía, se genera un mensaje de error, y no se efectúa ninguna transferencia , pudiendo adoptarse, como en la GOSUB, la decisión que se estime oportuna.

- CHECK : Verifica que el carácter en curso es el esperado ; si esta condición es cierta, avanza al carácter siguiente, llamando a un subprograma NEXTC que debe ser suministrado por el programador, y que proporcione el siguiente elemento a analizar ; si la condición es falsa, se genera un mensaje de error que se transmite a un subprograma del usuario, el cual puede tomar la decisión que estime conveniente (parar el análisis, dar símbolo por sobreentendido, etc.).

- FAULT : Genera la señal de error, activando un indicador con la clave del mismo. La rutina de recuperación de errores del usuario puede informarse

de qué error se trata y obrar en consecuencia.

- DECID : Elige entre continuar o efectuar una ruptura de control, según pertenezca el símbolo en curso - al conjunto propuesto ó no, y según sea la condición especificada, materializando la instrucción DECIDE.

5.8.2. Subprogramas de pilas.

Un sencillo conjunto de subprogramas permiten manejar un conjunto entero monodimensional como una pila de datos. Se han implementado dos pares de subprogramas que permiten almacenar en la pila - datos que pueden ser caracteres simples (palabras enteras) o cadenas de los mismos, con indicación del número de elementos que la componen.

5.8.3. Subprogramas de tratamiento de matrices booleanas.

Este paquete de subprogramas permite almacenar y procesar matrices booleanas compactadas, usando un bit de cada palabra para cada punto de la matriz, efectuándose el empaquetado por filas independientes. Los subprogramas implementados permiten : (36)

- INMBT : Iniciar a ceros la matriz completa.
- STBTM : Poner a 1 un punto de la matriz.
- CLBTM : Poner a 0 un punto de la matriz.
- LBTMT : Obtener el valor de un punto de la matriz.

- UNION : Hallar la unión de dos filas de la matriz, sustituyendo el resultado a una de ellas.
- CTRNR : Hallar el cierre transitivo de la matriz, según el algoritmo de Warshall. (37)
- WRMBT : Escribir una fila ó una columna de la matriz booleana, o la matriz completa por filas.

5.8.4. Subprogramas de solapes entre módulos.

Un par de subprogramas, o por mejor decir, un subprograma con dos puntos de entrada, OVLAY y OVRTN , diseñados al efecto, permiten solventar la dificultad que plantea el sistema operativo RTE-III al impedir que programas escritos en FORTRAN IV ó ALGOL puedan llamar segmentos en área de solape de manera que éstos sean capaces de devolver el control al punto en que se realizó la llamada. El sistema desarrollado es capaz de transmitir parámetros opcionales al segmento, y devolver al programa principal los resultados que éste proporcione, usando llamadas estándar del ejecutivo. (38),(39).

5.8.5. Subprograma de manejo de caracteres.

Como ayuda al sistema, y principalmente destinado al analizador lexicográfico, se ha desarrollado y puesto a punto una biblioteca de subprogramas para el tratamiento de cadenas de caracteres. Esta biblioteca maneja caracteres empaquetados (formato A2 del FORTRAN), y caracteres sueltos, en el llamado formato R 1, que consiste en colocar el carácter en los 8 bits menos significativos, dejando a cero -

los más significativos, forma en la que suelen suministrarlos la mayoría de los equipos de lectura/escritura. Para hacerlo compatible con su procesamiento FORTRAN IV usando formatos A, se han escrito algunas funciones auxiliares.

Los subprogramas de esta biblioteca son : (40)

- IGTBT : obtener un carácter de una cadena, indicando la posición del mismo.
- STOBT : escribir un carácter en una cadena, en una posición elegida.
- MOVBT : Copiar una cadena de caracteres empaquetados o una subcadena de la misma en otra posición o en otra cadena.
- ICMBT : Comparar dos cadenas o subcadenas de caracteres empaquetados , indicando, según el orden alfabético si son iguales o distintas, y en este último caso, cual es la anterior y cual la posterior.
- ISCBT : Permite localizar un carácter en una cadena, buscándolo desde la posición inicial que se le señala hasta hallarlo, o encontrar otro carácter que marque el final de la misma.

5.8.6. Funciones para el tratamiento de bits.

Para suplir la falta en el FORTRAN de las posibilidades de desplazamientos de bits, que son disponibles a nivel de ensambla-

dor, se han escrito igualmente dos subprogramas SLA y SRA que realizan los desplazamientos lógicos, a la izquierda y a la derecha, del dato suministrado, con indicación del número de bits a desplazar.

No se han escrito subprogramas para las operaciones de unión, intersección y unión exclusiva, debido a que el compilador de FORTRAN IV usado dispone de funciones intrínsecamente definidas que realizan estas composiciones binarias.

5.9. Programa MCMCX.

Este es un programa auxiliar que permite enlazar todos los programas principales del sistema , así como el ensamblador del RTE-III, diversificando la entrada de datos y la salida de resultados para que aquellos puedan estar en tarjetas, cintas, discos, etc, y que estos puedan obtenerse bien impresos, bien en fichero magnético ó perforados. Esta diversificación se consigue por llamdas al " SPOOL Monitor Program " del sistema operativo. (41)

CAPITULO VI

APLICACIONES REALIZADAS DEL GENERADOR AUTOMATICO DE ANALIZADORES

CAPITULO VI

APLICACIONES REALIZADAS DEL GENERADOR AUTOMATICO DE ANALIZADORES

6.1. Traductor de expresiones aritméticas a notación polaca postfija.

Como primer ejemplo sencillo, se ha implementado un programa que convierte expresiones escritas en notación aritmética a notación polaca postfija, generando el analizador de forma automática, y que por su estructura no necesita emplear ninguna pila de datos para la transformación. (37)

La gramática de las expresiones, con acciones incluídas - resulta :

```
FR  -> ES ";"

ES -> ( ["+"] T , "-" T 'MUNIT' ) [ RE ]

RE -> ("+" T 'SUMA', "-" T 'RESTA')+

T ->  F ["*" F 'MULT' , "/" F 'DIVIS']+

F ->  ( "I" 'IDENT' , "(" ES ")" ) [ "↑" F 'POT']
```

Las metavariables representan, respectivamente :

- FR : frase, una expresión terminada en punto y coma.
- ES : Expresión con posible signo inicial
- RE : resto de la expresión.
- T : término
- F : factor

El símbolo terminal I representa un identificador válido. En la gramática puede apreciarse que se asume, para el caso de potencias iterativas la forma más generalizada en FORTRAN IV :

$$A \uparrow B \uparrow C \equiv A \uparrow (B \uparrow C) \Rightarrow A\ B\ C\ \uparrow\uparrow$$

Si se quisiera adoptar la forma común en ALGOL, bastaría sustituir la última regla por

F --> F1 ["↑" F1 ' POT'] +

F1 --> " I " 'IDENT' , "(" ES ")"

Las acciones asociadas son :

- MUNIT : Cambiar de signo (sacar @).
- SUMA : Sacar + .
- RESTA : Sacar − .
- MULT : Sacar * .
- DIVIS : Sacar /.
- IDENT : Sacar identificador.
- POT : Sacar ↑ .

Como queda de manifiesto, es fácil modificar la semántica de la frase, alterando algunas reglas, sin necesidad, en este caso, de cambiar las acciones. Por otra parte, las formas de derivación recurrentes por la izquierda, se sustituyen fácilmente por formas iterativas, que, además de adaptarse a la condición 1 del determinismo descendente, conservan íntegramente el significado de la frase, lo cual no puede conseguirse sustituyendo una gramática recurrente por la izquierda por otra recurrente por la derecha, única opción presentada por la notación BNF, que solo determina una gramática débilmente equivalente, en tanto que con SRL se obtiene una gramática semánticamente equivalente.

Los listados de los programas y algunas de las pruebas realizadas a este convertidor se incorporan como apéndice. El analizador ha sido generado como subprograma, pudiendo incorporarse, por ejemplo, a un compilador incremental para BASIC ó FORTRAN, con pequeñas variaciones. Se ha escrito un programa que emplea este traductor para probar su eficacia, y el tiempo de respuesta obtenido es muy inferior a los tiempos muertos originados por los procesos de entrada/salida, dando la solución de manera prácticamente instantánea, aun con expresiones complicadas con varios niveles de paréntesis.

6.2. Gramática determinista de expresiones aritméticas y booleanas.

En ALGOL y otros lenguajes del mismo corte se presenta con frecuencia el problema de definir una gramática de expresiones que englobe tanto las operaciones aritméticas, como las operaciones

booleanas de relación, así como asignaciones de resultados parciales , en un contexto de análisis descendente, ya que para el análisis ascendente, la gramática de este tipo de expresiones está resuelta. Por ello se ha trabajado en la búsqueda de una gramática en notación BRL que defina dichas expresiones y además cumpla las condiciones de determinismo descendente LL(1). Tras un proceso laborioso de factorizaciones, reducciones, etc., se ha llegado a una gramática que cumple las condiciones propuestas. En esta aplicación no se han incorporado las acciones dentro de la descripción de la gramática del lenguaje, puesto que solo se trataba de obtener un analizador determinístico del mismo y no un traductor ; si se quiere obtener un convertidor a notación polaca, por ejemplo, bastará insertar acciones tan sencillas como las del caso anterior , en posiciones muy localizadas de las reglas, pudiendo añadirse, si se desea, rutinas de verificación de concordancia de tipos reales y enteros, o, como es frecuente, dejar este punto para el paso posterior de generación de código objeto.

La gramática obtenida, así como el analizador generado automáticamente pueden consultarse en el apéndice 2.

6.3. Analizador sintáctico para una variante del ALGOL.

Tomando como base la descripción formal del ALGOL 60 (42), (43) , unida a unas variaciones introducidas por el HP-ALGOL (44), que acepta las formas básicas de la programación estructurada, se ha desarrollado una gramática que describe programas en este lenguaje, y que cumple las condiciones del determinismo descendente, habiendose obte-

nido el analizador sintáctico automático correspondiente. Con base en esta aplicación, introduciendo algunas instrucciones adicionales para el tratamiento de cadenas de caracteres, se espera continuar el trabajo desarrollado, obteniendo en primer lugar un compilador completo para este lenguaje, que sea operativo sobre el sistema H.P. 21-MX disponible, y seguidamente, usando este lenguaje, re-escribir el generador de analizadores, optimizándolo en todos los puntos posibles, con lo que se espera obtener un aumento de la capacidad del mismo, estimándose un aumento del tiempo de proceso no superior al 25 % , sin que esto afecte en absoluto a la velocidad del analizador generado ; por otra parte, el aumento anterior puede compensarse fácilmente realizando algunas optimizaciones adicionales tanto en las implementaciones de los algoritmos de verificación de determinismo descendente, como en los pasos finales de generación del analizador.

La gramática desarrollada, así como el analizador producido automáticamente , pueden consultarse en el apéndice 3.

6.4. Consideraciones sobre tiempos de ejecución.

Se han realizado medidas de tiempo de ejecución del generador de analizadores implementado, habiéndose obtenido los resultados siguientes :

a) Generación del traductor de notación aritmética a polaca :

- Tiempo de CPU con generación incluída : 4.97 segundos
- Tiempo de CPU, con generación y traducción a ensambla

dor del HP 21MX : 9.28 segundos

- Tiempos totales, incluyendo procesos de lectura/escritura : 23.76 y 30.65 segundos respectivamente.

b) Aplicación a la variante del ALGOL de 6.3 :

- Tiempo de CPU con generación : 49.10 seg.
- Tiempo de CPU, con generación y traducción del analizador a ensamblador del HP 21MX : 1 min 26.76 seg.
- Tiempos totales : 1 min , 23.45 seg y 2 min, 17.98 seg.

Los tiempos totales indicados están en realidad aumentados en parte por las operaciones propias del sistema operativo, controles de entrada/salida adicionales, etc., y están obtenidos en procesos que realizan los listados opcionales mas frecuentes, lo que hace aumentar de paso el tiempo de CPU debido a las operaciones de formateado de resultados.

Como medida de la bondad del sistema desarrollado puede indicarse que el procesador del sistema Griffiths-Peltier, con una gramática de ALGOL aun más reducida que la indicada en 6.3, partiendo de la notación BNF, generó un analizador, sin traducirlo a código ensamblador, tomando un tiempo de 1.33 minutos en un IBM 360/67J ; esta medida es, lógicamente, solo una estimación, por tratarse de máquinas diferentes, con diferente ciclo básico, diferente sistema operativo, etc..

CAPITULO VII

CONCLUSIONES

CAPITULO VII

CONCLUSIONES

Se ha realizado un trabajo que comporta tanto aspectos teóricos como prácticos en el campo de la generación automática de analizadores sintácticos para compiladores .

En el transcurso del trabajo se han adaptado las condiciones de determinismo descendente de la máquina analítica de Knuth, ampliándolas a las gramáticas escritas en notación SRL ; se han desarrollado igualmente los algoritmos necesarios para la verificación de las citadas condiciones, y se han introducido mejoras semánticas y sintácticas en el lenguaje de producción de analizadores del sistema Griffiths-Peltier.

Igualmente, se ha implementado un completo sistema de programas que, para una gramática dada, efectúa la verificación de las condiciones de determinismo descendente, y si éstas se cumplen , genera automáticamente un analizador sintáctico para la citada gramática. Aunque se ha elegido el modelo de proceso por compilador de compiladores, puede convertirse fácilmente al modelo de metacompilador.

Se ha implementado también un procesador de macroinstrucciones del lenguaje de generación, específico para el computador

HP 21MX, pero que puede ser transformado fácilmente, adaptándolo a otro computador cualquiera.

La escritura de los programas se ha realizado en FORTRAN - IV principalmente, por razones de transportabilidad, empleando en numerosas partes la propia filosofía del analizador automático, basado en la gramática de la regla a procesar, empleando, por consiguiente, las bases de la autocompilación o ' bootstrapping ' .

Como resultado marginal, varios de los conjuntos o bibliotecas de subprogramas desarrollados para el sistema de programas pueden considerarse como de utilidad general, habiendo sido aplicados ya por diversos usuarios.

El sistema de programas ha sido aplicado en varios casos , en el seno de este trabajo, para probar la capacidad del mismo, su velocidad, y la comodidad de manejo, obteniéndose resultados que demuestran su potencia, ya expuestos anteriormente.

En cuanto repercusiones futuras de este trabajo, en la actualidad se consideran, de forma inmediata, los proyectos siguientes.

a) Compilador para lenguaje DYNAMO para el HP 21MX de este centro, adaptando el ya existente en el IBM 1130 (45) , y aplicando los resultados de este trabajo. Este proyecto está asignado ya como trabajo de fin de carrera.

b) Realización de un compilador de ALGOL 60, añadiéndole las formas de la programación estructurada, completamente desarrollado, con asignación dinámica de memoria.

c) Adaptación del procesador a su funcionamiento como meta compilador.

d) Diseño e implementación de un programa complementario de optimización de la gramática de entrada, tendente a su reducción en espacio ocupado y tiempo de ejecución.

e) Diseño e implementación de los algoritmos de Foster (46) y de Bordier (22),(47) , adaptados a la gramática en SRL , para la resolución de los problemas de transformación a la forma determinista descendente, dentro de las limitaciones lógicas de calculabilidad y decisibilidad.

f) Diseño de un lenguaje de simulación para el estudio y modelización regional , siguiendo las técnicas de la Dinámica de Sistemas, sobre la base del lenguaje APL, que comportaría definición modular y estratificada de modelos, acceso a bases de datos , etc.

Junto a estos trabajos de índole académica, existen firmas comerciales interesadas en la aplicación del sistema a la realización de compiladores adicionales para sus equipos, desarrollando de esta forma partes del sofware de base en España, buscando una economía en el mismo.

REFERENCIAS

1. GINSBURG, S.

"The mathematical theory of context free languages".
Mc Graw Hill, 1966.

2. ARBIB, M.

"Algebraic theory of machines, languages and semigroups".
Academy Press, 1968.

3. ARBIB, M.

"Theories of abstract automata"
Prentice - Hall, 1969.

4. HOPCROFT, J.E. + ULLMAN, J.D.

"Formal languages and their relation to automata"
Addison - Wesley, 1969.

5. KAIN, R.

"Automata theory. Machines and languejes".
Mac Graw Hill, 1972.

6. HOPGOOD, F.
"Compiling techniques ".
Mac Donald, 1969.

7. GRIES, D.

"Compiler construction for digital computers ".
Wiley, 1971 .

8. AHO, A.V. - ULLMAN, J. D.

"The theory of parsing, translation and compiling ".
Prentice Hall, 1972.

9. FLOYD, R.

"The syntax of programming languages - A survey ",en Rosen, "Programming system and languages".
Mc Graw Hill, 1967.

10. GRAHAM, R.

"Bounded Context translation".
en Rosen, Programming system & languages".
Mc Graw Hill, 1967.

11. GREIBACH, S. A.

"Formal parsing systems ".
C. ACM, 1964.

12. TERRINE, G.

"Construction automatique d'analyseurs syntaxiques ascendants deterministes a partir de C.F. grammaires eventuellement de contexte borné".
Tesis. Univ. Grenoble, 1972 .

13. ABRAMSON , H.

"Theory and aplication of a bottom-up sintax-directed translator."
Ac. Press - 1973 .

14. BROOKER, R.A. - MORRIS, D.

"A general translation program for phrase structure languages".
J. ACM . Enero 1962.

15. BROOKER - MORRIS

"The compiler compiler".
Annual Review of Automatic Programming. 1963. Vol. 3,

16. ROSEN, S.

"A compiler building system developed by Brooker and Morris ".
C. ACM, Julio 1964.

17. BROOKER, MORRIS, ROHL.

"Experience with the compiler - compiler"
Computer Journal. vol. 9, no. 4 - 1967.

18. GRIFFITHS, M.

"Analyse deterministe et compilateurs".
Tesis de Docteur d'etat.
Fac. des Sciences, Grenoble, 1969.

19. KOSTER, C.H.A.

"A compiler compiler"
Stichting Mathematisch Centrum, 1971.

20. CHEATHAN, T. - SATTLEY, K

"Syntax - directed compiling"
en Rosen, Programming system & languages,
Mc Graw-Hill, 1967.

21. WARSHALL, S. - SHAPIRO, R.

"A general purpose table-driver compiler"
en Rosen, Programming System and languages,
Mc Graw Hill, 1967.

22. BORDIER, J.

"Methodes pour la mise an point de grammaires LL(1) ".
Tesis, Grenoble, 1971.

23. KNUTH, D.E.
Top - Down syntax analysis.
International Summer School on Computer Programming.
Copenhagen 1967.

24. URSCHLER, G.

"Concrete syntax of PL/1 ".
IBM -- Viena / TR.25. 096, 1969.

25. ASSABGUI, M.

"Notation SRL et generation automatique d'analyseurs ".
N.T. 5359 bis. I.M.A. Grenoble, 1970 .

26. BORDIER, J.

"Notation ULD et analyse descendante deterministe."
C.C. IBM – Grenoble (1972)

27. ROVAYO, M.

"Lenguajes deterministicos y compiladores por análisis descendente."
Com. II Jornadas de Automática. E.T.S.I.I. Madrid, 1978.

28. GRIFFITHS, M. – PELTIER, M.

"A macro generable language for the 360 computers."
Computer Bulletin, vol. 13, nº 11 – 1969.

29. PELTIER, M.

"The Macro system"
Centro científico IBM – Francia
Estudio FF 2.0057.0, 1968.

30. RAMOS, I.

"Lenguajes para la estructura de traductores."
Congreso de Automática 72.

31. GRIFFITHS. M.

"LL(1) grammars and analysers.
Advanced course on compiler construction."
Universidad técnica de Munich, 1974.

32. DAHL, DIJKSTRA, HOARE

"Structured programming"
Academic Press, 1972.

33. FOSTER, J.M.

"A syntax improving program".
R.R.E. 1967.

34. FOSTER, J.M.

" A syntax improving device".
Comp. Jour. Mayo 1968.

35. CARVALLO, M.

"Logique à trois valeurs. Logique à seuil."
Gauthier - Villars, 1968.

36. ROVAYO, M.
"Subprogramas de tratamiento de matrices booleanas. Memoria de utilización".
Memoria interna. E,T.S.I.S. 1977.

37. WARSHALL, S.

"A theorem on boolean matrices"
J. ACM enero 1962.

38. ROVAYO, M.

"Subprogramas para llamada y recuperación de segmentos para el HP21MX bajo. RTE II/III ".
Memoria interna, E.T.S.I.I.S. 1977.

39. Real time Executive III. Software system programming and operating manual.
H.P. 92060 - 90004

40. ROVAYO, M.

"Biblioteca de subprogramas para manipulación de octetos en el HP 21MX desde FORTRAN o ALGOL."
Memoria interna, E.T.S.I.I.S. 1976.

41. Batch - Spool Monitor. Reference Manual
H.P. 92060 - 90013

42. BACKUS, et al.

"Revised report on the algorithmic language ALGOL 60."
C. ACM, vol. 6. Dic. 1960

43. ROSEN, S.

"The ALGOL programming language", en Rosen, "Prog. sys. & lang."
Mc Graw Hill 1967

44. HP. ALGOL. Reference Manual
H.P. 2116 - 9072

45. ROVAYO, M.

"Compilador DYNAMO para IBM 1130"
E.T.S.I.I.S - I.D.R. 1976

46. FOSTER, J.M.
Automatic syntactic analysis
Mac Donald - 1970

47. BORDIER, J.

"Elimination de la recursivité à gauche dans une grammaire context-free"
N.T. I.M.A. Grenoble, 1970

ANEXO A : MANUAL DE UTILIZACION DE LOS PROGRAMAS REALIZADOS.

ANEXO A : MANUAL DE UTILIZACION DE LOS PROGRAMAS REALIZADOS.

1. Preparación de los datos de la gramática.

El paquete de datos que constituyen la definición de la gramática consta de tres partes bien diferenciadas :

a) Tarjeta de control: señala las opciones de proceso escogidas entre las posibles, y el nombre que se desea dar al analizador generado.

b) Tarjetas de definición del vocabulario, que pueden omitirse si se desea.

c) Tarjetas de reglas de producción.

Adicionalmente, tras la tarjeta de control pueden insertarse comentarios de dos formas distintas:

a) Mediante tarjeta de comentario, colocando un asterisco (*) en la primera columna.

b) Al final de cada regla de producción, dentro de la misma linea en que termine.

Aunque en todo el texto se hagan referencia a "tarjetas", esta debe interpretarse como un registro, del tipo que sea, con un máximo de 72 caracteres.

1.1. Tarjeta de control.

Comienza en la primera columna, y debe escribirse sin espacios intercalados. El formato de esta tarjeta es el siguiente:

MOMP, opciones "nombre asignado" -.

a) Los simbolos subrayados deben aparecer como estan escritos.

b) Las opciones aludidas tienen la forma de una estructura iterativa: [opción ,]+

Las opciones elegidas pueden aparecer en cualquier orden, siendo las claves posibles las que siguen:

- L : Listar tarjetas de entrada de la gramática.
- M : Listar analizador en lenguaje de generación (lenguaje de macros).
- T : Listar tabla de símbolos generada por el procesador de reglas.
- E : Listado de las operaciones y cálculos intermedios realizados. Esta función está destinada principalmente a la puesta a punto de los programas, y no es necesaria para los usuarios del sistema.
- A : Listado del programa generado, en lenguaje ensamblador del HP 21MX.

- S : El analizador generado deberá construirse como un subprograma, con llamada compatible con FORTRAN y ALGOL del HP 21MX. Por omisión, el analizador se construye como programa principal.

- B : Perforación del analizador generado, tras su traducción a lenguaje ensamblador.

- C : Impresión de tablas binarias de derivación en la cadena vacía y matrices de relaciones binarias de siguiente, final y vecino.

c) El nombre asignado al analizador debe ser un nombre de programa o subprograma válido para el HP 21MX, esto es, un máximo de cinco caracteres alfanuméricos, siendo alfabético el primero de ellos. En esta consideración de alfabéticos se incluyen algunos caracteres especiales, como por ejemplo: ¢, . , @ , y otros.

d) Ejemplo de tarjeta de control :

MCMP, L, T, B, S, " ANAL"

1.2. Tarjetas de definición de vocabulario.

Este bloque tiene la forma,

VOCABULARIO : vocabulario ;

El "vocabulario" indicado por la definición anterior puede contener tanto nombres de clase, como simbolos básicos entre comillas, admitiéndose también las funciones semánticas entre apóstrofes.- Cada uno de estos elementos del vocabulario debe ir separado del siguiente por una coma, admitiéndose espacios en blanco en cualquier parte, excepto en el interior de los nombres de clase y de las funciones. En caso de no caber en una sola tarjeta, puede continuarse en tantas como se necesiten, marcando el final de la lista con un punto y coma.

La misión fundamental de esta definición previa del vocabulario es permitir al usuario conocer de antemano y prefijar el valor numérico que el procesador va a asignar a cada uno de los elementos del mismo, para no tener que cambiar el analizador léxicográfico cada vez que una redefinición de reglas altere el orden en que aparezcan los símbolos básicos en la gramática.

Por otra parte, el primer nombre de clase que aparezca, bien en esta lista de vocabulario, bien en la primera regla, si no se ha definido previamente, se toma como axioma de la gramática .

Tras el punto y coma que termina la lista pueden insertarse comentarios hasta el final de la tarjeta.

1.3. Tarjetas de reglas de producción.

El paquete de tarjetas con las reglas de producción se prepara como sigue :

a) una tarjeta con PRODUCCIONES: comenzando en la columna 1.

b) El conjunto de reglas de la gramática, en el formato que se indica más abajo.

c) Una tarjeta de final de reglas con el simbolo $\emptyset$ en la columna 1.

Las reglas se escriben en formato libre, sin blancos en el interior de los nombres de clase y de las funciones, y respetando la forma en que se hubiera descrito en las tarjetas de vocabulario. La forma general es :

nombre de clase ⟶ definición ;

El simbolo de derivación puede escribirse como -> (guión, mayor que) o como => (igual, mayor que). La definición puede ocupar tantas líneas como sea necesario o como se desee, terminando en un punto y coma. El resto, del punto y coma hasta el final de la línea, puede emplearse para comentario. Pueden usarse espacios a discreción entre elementos para facilitar la lectura y comprensión de las reglas. Debe tenerse en cuenta que cada regla tiene que comenzar en una línea nueva.

2. Ejecución del programa MCMP.

El programa MCMP se ejecuta con la orden

RU, MCMP, e, i, f_1, f_2, f_3

con los parámetros que a continuación se describen :

- e : unidad de entrada de datos de la gramática.
- i : unidad impresora.
- f_1, f_2, f_3 : unidades de cinta magnética de trabajo. El dispositivo f_1 retiene una copia de la forma intermedia de la gramática, y en f_3 se obtiene el analizador escrito en lenguaje de generación.

Como ya se ha dicho, el programa MCMP enlaza automaticamente con el PCLL1, que verifica las condiciones de determinismo, y con el MCGEN, que genera el analizador en lenguaje de macros.

Por omisión de los parámetros se toma :

e = 5 (lectora del sistema)
i = 6 (impresora del sistema)
f_1 = 61
f_2 = 62
f_3 = 63

En caso de no disponer de unidades de cinta magnética, como ocurre en la instalación en que se ha implementado, se puede hacer uso de la asignación de unidades lógicas a ficheros en disco, mediante el proceso en tanda, bajo control del File Manager (FMGR) del sistema.

3. Ejecución del programa PMMC.

La orden de ejecución de PMMC es la siguiente :

RU, PMMC, e , i , p , n

siendo :

e - unidad de entrada en la que está el analizador en lenguaje de generación (o de macros).

i - unidad impresora en la que se obtiene el listado del analizador en macros, si se solicita.

p - unidad perforadora (o de cinta magnética o similar) en la que el programa PMMC genera el equivalente en ensamblador del analizador de entrada.

n - número de líneas por página que admite el formulario de la impresora.

Alguno, o incluso todos los parámetros pueden omitirse, tomando por defecto los valores siguientes:

e = 5 (unidad de lectura del sistema).

i = 6 (impresora del sistema).

p = 4 (perforadora del sistema)

n = 56 (número de líneas útiles tras descontar la cabecera de página en papel normal de 11 pulgadas, con impresión de 6 líneas por pulgada).

La ejecución de este programa se ha dejado independiente del resto, facilitando así los cambios, en caso de que se desee generar código de ensamblador de otra máquina, o bien generar unas tablas de datos para un meta compilador.

En caso de que algunas de las unidades sea un fichero, la equivalencia puede realizarse trabajando en tanda, bajo el control del FMGR (File Manager) del sistema de explotación.

4. Ejecución combinada, mediante el programa MCMPX.

Como ya se ha expuesto, el programa MCMPX controla la ejecución de los dos anteriores, generando unidades lógicas de entrada/salida ficticias, mediante llamadas al Spool Monitor Program (SMP) del sistema RTE del HP 21MX, creando ficheros intermedios que simulan las unidades de cinta magnética, evitando el trabajo en tanda.

Los parámetros de la ejecución de MCMPX son :

RU , MCMPX c , i , am , ae

representando :

c : unidad de consola en que se desean recibir los mensajes de error de MCMPX, en caso de existir.

g : unidad de entrada, o nombre del fichero, en que se encuentran los datos de la gramática.

i : unidad de salida impresa.

am : unidad de salida o fichero en el cual se desea obtener el analizador, en lenguaje de generación.

ae : unidad de salida o fichero en el cual se puede obtener el analizador, traducido a un programa o subprograma en lenguaje ensamblador del HP 21MX. Si se omite, no se realiza esta traducción, suprimiéndose la ejecución del programa PMMC.

5. Formato del analizador en lenguaje de generación, para el programa PMMC.

Aunque el usuario normalmente no tiene necesidad de conocer este formato, se incorpora aquí una breve descripción del mismo, destinada a aquellos usuarios especializados que deseen construir su propio programa de traducción, para otro equipo, o bien programar el analizador directamente en el lenguaje de generación, ya sea - para incorporar una especificación nueva ro soportada, ya sea por economía, tratando de optimizar el código resultante más de lo que el sistema es capaz, así como por cualquier otro motivo.

El paquete de datos para el programa PMMC está formado por :

a) Una tarjeta de control, que selecciona las opciones posibles, e indica el nombre del axioma y el que debe asignarse al analizador resultante.

b) Las instrucciones del analizador, en el lenguaje de generación.

c) Línea de fin de analizador.

5.1. Tarjeta de control.

Tiene la forma siguiente, comenzando en la columna 1, y sin espacios intercalados.

PMMC, [opciones] "nombre de analizador" : axioma.

Las opciones tienen un formato igual que en el caso del programa MCMP: lista de opciones separadas por comas, elegidas en orden arbitrario de entre las posibilidades siguientes :

- L : Listar fuente del analizador en macros.
- A : Listar fuente del analizador en lenguaje ensamblador.
- B : Perforar programa traducido a ensamblador.
- S : Generar analizador en modo subprograma.

5.2. Formato de las instrucciones del analizador, en lenguaje de generación.

El formato de las instrucciones es el siguiente :

columnas 1 a 5 : etiqueta de la instrucción. El programa PMMC dedica poca atención a las mismas, por lo que deben ser etiquetas válidas del ensamblador - del HP21MX.

columna 6 : en blanco.

columnas 7 a 12: código de la instrucción, ajustado a la izquierda.

columna 13 : en blanco.

columnas 14 a 72: operando sin espacios intercalados. Tras el - primer espacio, el resto se considera como un comentario.

En las instrucciones DECIDE, en las cuales la lista de decisión puede ser larga y exceder del campo utilizable, se admite la continuación en lineas sucesivas, marcando con un apóstrofe la columna 73 de la línea que es continuada, y comenzando la siguiente en la columna 1.

5.3. Línea de fin de programa.

Es, sencillamente, la palabra END en el campo de instrucción. Esta instrucción no genera código ejecutable, y solamente señala el fin del programa analizador, indicando a PMMC que ya no hay más instrucciones a traducir.

ANEXO B : CONVERTIDOR DE NOTACION ARITMETICA A POLACA POSTFIJA.

```
JOB GANEXP     ON AT 13:49:53.82  ON 24 AUG  1978


MCMP,L,M,A,T,S,B,"ANEXP"

002  VOCABULARIO: "+","-","*","/","^","(",")",";","I" ;
003  PRODUCCIONES:
004  FR=> ES ";" ;
005  ES=> ( ["+"] T , "-" T 'MUNIT' ) [ RE ] ;
006  RE=> ("+" T 'SUMA', "-" T 'RESTA')+ ;
007  T=>  F ["*" F 'MULT' , "/" F 'DIVIS']+ ;
008  F=>  ( "I" 'IDENT' , "(" ES ")")[ "^" F 'POT'I  ;
009  $


 ** TABLA DE SIMBOLOS **


 N. CLASES

  1  FR       2  ES       3  T        4  RE       5  F


 S. BASICOS

  1  +        2  -        3  *        4  /        5  ^
  6  (        7  )        8  ;        9  I


 FUNCIONES

  1  MUNIT    2  SUMA     3  RESTA    4  MULT     5  DIVIS
  6  IDENT    7  POT

LA GRAMATICA CONSTA DE   5 NOMBRES DE CLASE,   9 SIMBOLOS BASICOS Y   7 FU
```

```
 1  PMMC,L,A,S,D,"ANEXP":FR
 2          ROUTIN FR
 3          GOSUB  ES
 4          CHECK  8
 5          RETURN
 6          ROUTIN ES
 7          DECIDE (9,6,1),.F.,@2
 8          DECIDE 1,.F.,@1
 9          CHECK  1
10  @1      GOSUB  T
11          GOTO   @4
12  @2      DECIDE 2,.F.,@3
13          CHECK  2
14          GOSUB  T
15          CALL   MUNIT
16          GOTO   @4
17  @3      FAULT  250
18  @4      DECIDE (2,1),.F.,@5
19          GOSUB  RE
20  @5      RETURN
2.          ROUTIN RE
22  @6      DECIDE 1,.F.,@7
23          CHECK  1
24          GOSUB  T
25          CALL   SUMA
26          GOTO   @9
27  @7      DECIDE 2,.F.,@8
28          CHECK  2
29          GOSUB  T
30          CALL   RESTA
31          GOTO   @9
32  @8      FAULT  260
33  @9      DECIDE (2,1),.T.,@6
34          RETURN
35          ROUTIN T
36          GOSUB  F
37  @10     DECIDE (4,3),.F.,@12
38          DECIDE 3,.F.,@11
39          CHECK  3
40          GOSUB  F
41          CALL   MULT
42          GOTO   @12
43  @11     CHECK  4
44          GOSUB  F
45          CALL   DIVIS
46          GOTO   @10
47  @12     RETURN
48          ROUTIN F
49          DECIDE 9,.F.,@13
50          CHECK  9
51          CALL   IDENT
52          GOTO   @15
53  @13     DECIDE 6,.F.,@14
54          CHECK  6
55          GOSUB  ES
56          CHECK  7
```

ANALIZADOR DEL TRADUCTOR,

EN LENGUAJE DE GENERACION.

```
 57          GOTO    @15
 58  @14     FAULT   261
 59  @15     DECIDE 5,.F.,@16
 60          CHECK   5
 61          GOSUB   F
 62          CALL    POT
 63  @16     RETURN
 64          END
 JOB GANEXP    OFF AT 13:50:41.69  ON 24 AUG  1978
EXECUTION TIME:  00:00:12.68
```

```
JOB CNARPL     ON AT 13:51:12.59  ON 24 AUG  1978


        PAGE  0001            FTN4 COMPILER: HP24177  (SEPT. 1974)


0001   FTN4,L
0002          PROGRAM NARPL,3
0003          COMMON LOG,IMPR,LARIT(35),LPOL(70),INARI,INPOL,IDNAM
0004          INTEGER IPARM(5)
0005   C
0006   C
0007          CALL RMPAR(IPARM)
0008          LOG=IPARM(1)
0009          IMPR=IPARM(2)
0010          IF(LOG  .LE. 0 .OR. LOG  .GT. 63) LOG =1
0011          IF(IMPR .LE. 0 .OR. IMPRE .GT. 63) IMPR=LOG
0012   10      WRITE(LOG,110)
0013   110     FORMAT("?_")
0014          READ(LOG,120)LARIT
0015   120     FORMAT(35A2)
0016          IF(LARIT(1) .EQ. 2H/*) CALL EXEC(6)
0017          INARI=1
0018          INPOL=0
0019          DO 130 I=1,70
0020   130     LPOL=2H
0021          IF(IMPR .EQ. LOG) GO TO 14
0022           WRITE(IMPR,140)LARIT
0023   140      FORMAT(/3X,35A2)
0024   14     CALL NEXTC
0025          CALL ANEXP
0026          WRITE(IMPR,150)(LPOL(I),I=1,INPOL)
0027   150     FORMAT(2X,65(1X,R1))
0028          GO TO 10
0029         END


**  NO ERRORS**     PROGRAM = 00175       COMMON = 00110
```

PROGRAMA DE PRUEBA.

```
0030          SUBROUTINE NEXTC
0031          COMMON LOG,IMPR,LARIT(35),LPCL(70),INARI,INPCL,IDNAM
0032          INTEGER LOPER(5)
0033          DATA LOPER/2H+-, 2H*/, 2H^(, 2H)), 2H  /
0034          IER=IFAUL(I)
0035          IF(IER)2,1,3
0036    C
0037     1    ICARC=IGTBT(LARIT,INARI)
0038          INARI=INARI+1
0039    C     BUSCAR CARACTER
0040          IOP=ISCBT(LOPER,1,ICARC,2H  )
0041          IF(IOP)11,11,12
0042     11    IF(ICARC .EQ. IAND(255,2H  )) GO TO 1
0043           IF(ICARC .GE. IAND(255,2H A) .AND. ICARC .LE. IAND(255,2H Z)
0044         1    110,13
0045    C     IDENTIFICADOR
0046     110    IDNAM=ICARC
0047           IOP=9
0048    12     CALL SETCR(IOP)
0049           RETURN
0050    C
0051     13    ICER=INARI-1
0052           WRITE(IMPR,130) ICER
0053    130     FORMAT(" CARACTER NO VALIDO EN COL."I3". SE IGNORA.")
0054           GO TO 1
0055    C
0056    C      ERRORES DEL PROCESO
0057     2    CONTINUE
0058     3     ICER=INARI-1
0059           WRITE(IMPR,310) IER,ICER
0060     310    FORMAT(" ERROR",I4," COL.",I3)
0061           RETURN
0062          END

**  NO ERRORS**     PROGRAM = 00163       COMMON = 00110
```

SUBPROGRAMA DE OBTENCION DEL CARACTER SIGUIENTE,

E IMPRESION DE ERRORES.

```
0063          SUBROUTINE MUNIT
0064          CALL ADPOL(2H 0)
0065          END
```

** NO ERRORS** PROGRAM = 00011 COMMON = 00000


```
0066          SUBROUTINE SUMA
0067          CALL ADPOL(2H +)
0068          END
```

** NO ERRORS** PROGRAM = 00012 COMMON = 00000


```
0069          SUBROUTINE RESTA
0070          CALL ADPOL(2H -)
0071          END
```

** NO ERRORS** PROGRAM = 00012 COMMON = 00000


```
0072          SUBROUTINE MULT
0073          CALL ADPOL(2H *)
0074          END
```

** NO ERRORS** PROGRAM = 00011 COMMON = 00000


```
0075          SUBROUTINE DIVIS
0076          CALL ADPOL(2H /)
0077          END
```

** NO ERRORS** PROGRAM = 00012 COMMON = 00000

```
        PAGE  0001          FTN4 COMPILER: HP24177  (SEPT. 1974)


0078          SUBROUTINE POT
0079          CALL ADPOL(2H ^)
0080          END

**  NO ERRORS**     PROGRAM = 00012       COMMON = 00000
```

```
        PAGE  0001          FTN4 COMPILER: HP24177  (SEPT. 1974)


0081          SUBROUTINE IDENT
0082          COMMON LOG,IMPR,LARIT(35),LPOL(70),INARI,INPOL,IDNAM
0083          CALL ADPOL(IDNAM)
0084          END

**  NO ERRORS**     PROGRAM = 00014       COMMON = 00110
```

```
        PAGE  0001          FTN4 COMPILER: HP24177  (SEPT. 1974)


0085          SUBROUTINE ADPOL(L)
0086          COMMON LOG,IMPR,LARIT(35),LPOL(70),INARI,INPOL,IDNAM
0087          INPOL=INPOL+1
0088          LPOL(INPOL)=L
0089          END

**  NO ERRORS**     PROGRAM = 00024       COMMON = 00110
```

ACCIONES

0001 ASMB,L
** NO ERRORS PASS#1 **RTE ASMB 760924**

ANALIZADOR DEL TRADUCTOR, EN

EN LENGUAJE ENSAMBLADOR.

```
0001                    ASMB,L
0002   00000                    NAM AKEXP,7
0003                            ENT AKEXP
0004                            EXT .ENTR
0005   00000 000000   AKEXP NOP
0006   00001 016001X            JSB .ENTR
0007   00002 000000R            DEF *-2
0008   00003 016002X            JSB GOSUB
0009   00004 000006R            DEF *+2
0010   00005 000007R            DEF FR
0011   00006 126000R            JMP AKEXP,I
0012                            EXT GOSUB,RETUR,CHECK,DECID,FAULT
0013*
0014****
0015*
0016   00007            FR      EQU *
0017   00007 016002X            JSB GOSUB
0018   00010 000012R            DEF *+2
0019   00011 000017R            DEF ES
0020   00012 016004X            JSB CHECK
0021   00013 000015R            DEF *+2
0022   00014 000010             DEC   8
0023   00015 016003X            JSB RETUR
0024   00016 000017R            DEF *+1
0025****
0026   00017            ES      EQU *
0027   00017 016005X            JSB DECID
0028   00020 000024R            DEF *+4
0029   00021 000025R            DEF *+4
0030   00022 000000             DEC 0
0031   00023 000045R            DEF @2
0032   00024 026027R            JMP *+003B
0033   00025 004406             OCT 004406
0034   00026 000400             OCT 000400
0035   00027 016005X            JSB DECID
0036   00030 000034R            DEF *+4
0037   00031 000035R            DEF *+4
0038   00032 000000             DEC 0
0039   00033 000041R            DEF @1
0040   00034 026036R            JMP *+002B
0041   00035 000400             OCT 000400
0042   00036 016004X            JSB CHECK
0043   00037 000041R            DEF *+2
0044   00040 000001             DEC   1
0045   00041 016002X @1         JSB GOSUB
0046   00042 000044R            DEF *+2
0047   00043 000162R            DEF T
0048   00044 026070R            JMP @4
0049   00045 016005X @2         JSB DECID
0050   00046 000052R            DEF *+4
0051   00047 000053R            DEF *+4
0052   00050 000000             DEC 0
0053   00051 000065R            DEF @3
0054   00052 026054R            JMP *+002B
0055   00053 001000             OCT 001000
0056   00054 016004X            JSB CHECK
```

```
0057   00055  000057R          DEF *+2
0058   00056  000002           DEC   2
0059   00057  016002X          JSB GOSUB
0060   00060  000062R          DEF *+2
0061   00061  000162R          DEF T
0062   00062  016007X          JSB MUNIT
0063   00063  000064R          DEF *+1
0064                           EXT MUNIT
0065   00064  026070R          JMP @4
0066   00065  016006X @3       JSB FAULT
0067   00066  000070R          DEF *+2
0068   00067  000402           DEC     258
0069   00070  016005X @4       JSB DECID
0070   00071  000075R          DEF *+4
0071   00072  000076R          DEF *+4
0072   00073  000000           DEC 0
0073   00074  000103R          DEF @5
0074   00075  026100R          JMP *+003B
0075   00076  001001           OCT 001001
0076   00077  000000           OCT 000000
0077   00100  016002X          JSB GOSUB
0078   00101  000107R          DEF *+2
0079   00102  000105R          DEF RE
0080   00103  016002X @5       JSB RETUR
0081   00104  000105R          DEF *+1
0082****
0083   00105          RE       EQU *
0084   00105  016005X @6       JSB DECID
0085   00106  000112R          DEF *+4
0086   00107  000113R          DEF *+4
0087   00110  000000           DEC 0
0088   00111  000125R          DEF @7
0089   00112  026114R          JMP *+002B
0090   00113  000400           OCT 000400
0091   00114  016004X          JSB CHECK
0092   00115  000117R          DEF *+2
0093   00116  000001           DEC   1
0094   00117  016002X          JSB GOSUB
0095   00120  000122R          DEF *+2
0096   00121  000162R          DEF T
0097   00122  016010X          JSB SUMA
0098   00123  000124R          DEF *+1
0099                           EXT SUMA
0100   00124  026150R          JMP @9
0101   00125  016005X @7       JSB DECID
0102   00126  000132R          DEF *+4
0103   00127  000133R          DEF *+4
0104   00130  000000           DEC 0
0105   00131  000145R          DEF @8
0106   00132  026134R          JMP *+002B
0107   00133  001000           OCT 001000
0108   00134  016004X          JSB CHECK
0109   00135  000137R          DEF *+2
0110   00136  000002           DEC   2
0111   00137  016002X          JSB GOSUB
0112   00140  000142R          DEF *+2
```

```
0113   00141 000162R          DEF T
0114   00142 016011X          JSB RESTA
0115   00143 000144R          DEF *+1
0116                          EXT RESTA
0117   00144 026150R          JMP @9
0118   00145 016006X @8       JSB FAULT
0119   00146 000150R          DEF *+2
0120   00147 000404           DEC     260
0121   00150 016005X @9       JSB DECID
0122   00151 000155R          DEF *+4
0123   00152 000156R          DEF *+4
0124   00153 000001           DEC 1
0125   00154 000165R          DEF @6
0126   00155 026160R          JMP *+003B
0127   00156 001001           OCT 001001
0128   00157 000000           OCT 000000
0129   00160 016003X          JSB RETUR
0130   00161 000162R          DEF *+1
0131****
0132   00162           T      EQU *
0133   00162 016002X          JSB GOSUB
0134   00163 000165R          DEF *+2
0135   00164 000230R          DEF F
0136   00165 016005X @10      JSB DECID
0137   00166 000172R          DEF *+4
0138   00167 000173R          DEF *+4
0139   00170 000000           DEC 0
0140   00171 000226R          DEF @12
0141   00172 026175R          JMP *+003B
0142   00173 002003           OCT 002003
0143   00174 000000           OCT 000000
0144   00175 016005X          JSB DECID
0145   00176 000202R          DEF *+4
0146   00177 000203R          DEF *+4
0147   00200 000000           DEC 0
0148   00201 000215R          DEF @11
0149   00202 026204R          JMP *+002B
0150   00203 001400           OCT 001400
0151   00204 016004X          JSB CHECK
0152   00205 000207R          DEF *+2
0153   00206 000003           DEC   3
0154   00207 016002X          JSB GOSUB
0155   00210 000212R          DEF *+2
0156   00211 000230R          DEF F
0157   00212 016012X          JSB MULT
0158   00213 000214R          DEF *+1
0159                          EXT MULT
0160   00214 026226R          JMP @12
0161   00215 016004X @11      JSB CHECK
0162   00216 000220R          DEF *+2
0163   00217 000004           DEC   4
0164   00220 016002X          JSB GOSUB
0165   00221 000223R          DEF *+2
0166   00222 000230R          DEF F
0167   00223 016013X          JSB DIVIS
0168   00224 000225R          DEF *+1
```

```
0169                                    EXT DIVIS
0170   00225 026165R                    JMP @10
0171   00226 016003X @12                JSB RETUR
0172   00227 000230R                    DEF *+1
0173****
0174   00230           F                EQU *
0175   00230 016005X                    JSB DECID
0176   00231 000235R                    DEF *+4
0177   00232 000236R                    DEF *+4
0178   00233 000000                     DEC 0
0179   00234 000245R                    DEF @13
0180   00235 026237R                    JMP *+002B
0181   00236 004400                     OCT 004400
0182   00237 016004X                    JSB CHECK
0183   00240 000242R                    DEF *+2
0184   00241 000011                     DEC   9
0185   00242 016014X                    JSB IDENT
0186   00243 000244R                    DEF *+1
0187                                    EXT IDENT
0188   00244 026271R                    JMP @15
0189   00245 016005X @13                JSB DECID
0190   00246 000252R                    DEF *+4
0191   00247 000253R                    DEF *+4
0192   00250 000000                     DEC 0
0193   00251 000266R                    DEF @14
0194   00252 026254R                    JMP *+002B
0195   00253 003000                     OCT 003000
0196   00254 016004X                    JSB CHECK
0197   00255 000257R                    DEF *+2
0198   00256 000006                     DEC   6
0199   00257 016002X                    JSB GOSUB
0200   00260 000262R                    DEF *+2
0201   00261 000017R                    DEF ES
0202   00262 016004X                    JSB CHECK
0203   00263 000265R                    DEF *+2
0204   00264 000007                     DEC   7
0205   00265 026271R                    JMP @15
0206   00266 016006X @14                JSB FAULT
0207   00267 000271R                    DEF *+2
0208   00270 000405                     DEC     261
0209   00271 016005X @15                JSB DECID
0210   00272 000276R                    DEF *+4
0211   00273 000277R                    DEF *+4
0212   00274 000000                     DEC 0
0213   00275 000310R                    DEF @16
0214   00276 026300R                    JMP *+002B
0215   00277 002400                     OCT 002400
0216   00300 016004X                    JSB CHECK
0217   00301 000303R                    DEF *+2
0218   00302 000005                     DEC   5
0219   00303 016002X                    JSB GOSUB
0220   00304 000306R                    DEF *+2
0221   00305 000230R                    DEF F
0222   00306 016015X                    JSB POT
0223   00307 000310R                    DEF *+1
0224                                    EXT POT
```

```
225   00310 016000ZX 816    JSB RETUR
226   00311 000312R         DEF *+1
227****
228                         END
*  NO ERRORS *TOTAL **RTE ASMB 760924**



/LOADR:HARPL READY
/LOADR:$END
JOB CHARPL    OFF AT 13:52:47.69  ON 24 AUG  1978
EXECUTION TIME:  00:00:20.29
```

```
A+B+C-D;
A B + C + D -



A-B-C-D+J;
A B - C - D - J +



A-B*C+(A/F^R)*(D-G);
A B C * - A F R ^ / D G - * +



(A+(((B))*C));
A B C * +



(((((((V)))))));
V



  A+(A*C+V
ERROR    7 COL.   9
ERROR    8 COL.   9
  A A C * V + +



  A+*V;
ERROR 261 COL.   3
  A V * +



  (-B+(B^2-4*A*C)^M)/(2*A)
CARACTER NO VALIDO EN COL.   8. SE IGNORA.
ERROR 261 COL.   9
CARACTER NO VALIDO EN COL.  10. SE IGNORA.
ERROR 261 COL.  11
ERROR    7 COL.  13
ERROR    8 COL.  18
  B @ B ^ A * - C * + M ^
```

PRUEBAS DEL PROGRAMA

ANEXO C : ANALIZADOR DE EXPRESIONES ARITMETICAS Y BOOLEANAS.

```
JOB ANEXP4     ON AT 13:45:37.38  ON 24 AUG  1978

MCMP,L,M,A,T,S,"ANEX4"

002  VOCABULARIO: "+", "-", "*", "/", "^", "(", ")", ":=", "I", "C" ;
003  PRODUCCIONES:
004  X1=> ID X2 ;
005   X2=> ":=" X21 ;
006    X21=> X3A, "-" X41 [X4] , IFC EX "ELSE" EX, "NOT" F1 [B5] [B4] ;
007     X3A=> ID ( X2, X3A1), (CTE, X31)  X3A1  ;
008      X3A1=> [X6] [X5] [X4] [B6] [B5] [B4] ;
009      X31=> "(" X21 ")" ;
010       X4=> (ADOP X41 )+ ;
011        ADOP-> "+", "-" ;
012        X41=> F1 [X5];
013       X5=> ( MULOP F1 )+ ;
014        MULOP-> "*" , "/" , "\" , "MOD" ;
015        F1=> F2 [X6] ;
016        F2=> ID, CTE, X31 ;
017       X6=> ( "^" F2 )+ ;
018       IFC-> "IF" EXBOOL "THEN" ;
019        EX-> F2 X3A1 ;
020        EXARIT-> F2 [X6] [X5] [X4] ;
021        EXBOOL-> EXARIT ( B6 [B5] [B4] , B5 [B4] , B4) ;
022      B4-> ( "OR" BF1 [B5])+ ;
023      B5-> ( "AND" BF1 )+ ;
024       BF1-> "NOT" F1 , F1 [B6] ;
025      B6-> OPREL F1;
026       OPREL-> "=" , "#" , ">=" , ">" , "<" , "<=" ;
027   ID=> "I" ;
028   CTE=> "C" ;
029  $
```

```
** TABLA DE SIMBOLOS **


N. CLASES

 1   X1       2   ID       3   X2       4   X21      5   X3A
 6   X41      7   X4       8   IFC      9   EX      10   F1
11   B5      12   B4      13   X3A1    14   CTE     15   X31
16   X6      17   X5      18   B6      19   ADOP    20   MULOP
21   F2      22   EXDOO   23   EXARI   24   BF1     25   OPREL


S. BASICOS

 1   +        2   -        3   *        4   /        5   ^
 6   (        7   )        8   :=       9   I       10   C
11   ELSE    12   NOT     13   \       14   MOD     15   IF
16   THEN    17   OR      18   AND     19   =       20   #
21   >=      22   >       23   <       24   <=


FUNCIONES


LA GRAMATICA CONSTA DE  25 NOMBRES DE CLASE,  24 SIMBOLOS BASICOS Y    0 F


 JOB ANEXP4    OFF AT 13:46:18.76  ON 24 AUG  1978
EXECUTION TIME:  00:00:16.53
```

EJEMPLO, SIN TRADUCCION A LENGUAJE ENSAMBLADOR.

```
JOB ANEXP4     ON AT 13:43:00.27  ON 24 AUG  1978

NCMP,L,M,A,T,S,"ANEX4"

002  VOCABULARIO: "+", "-", "*", "/", "^", "(", ")", ":=", "I", "C" ;
003  PRODUCCIONES:
004  X1=> ID X2 ;
005   X2=> ":=" X21 ;
006    X21=> X3A, "-" X41 [X4] , IFC EX "ELSE" EX, "NOT" F1 [B5] [B4] ;
007     X3A=> ID ( X2, X3A1), (CTE, X31)  X3A1  ;
008      X3A1=> [X6] [X5] [X4] [B6] [B5] [B4] ;
009      X31=> "(" X21 ")" ;
010       X4=> (ADOP X41 )+ ;
011        ADOP-> "+", "-" ;
012        X41=> F1 [X5];
013       X5=> ( MULOP F1 )+ ;
014        MULOP-> "*" , "/" , "\" , "MOD" ;
015        F1=> F2 [X6] ;
016        F2=> ID, CTE, X31 ;
017       X6=> ( "^" F2 )+ ;
018       IFC-> "IF" EXBOOL "THEN" ;
019        EX-> F2 X3A1 ;
020        EXARIT-> F2 [X6] [X5] [X4] ;
021        EXBOOL-> EXARIT ( B6 [B5] [B4] , B5 [B4] , B4) ;
022      B4-> ( "OR" BF1 [B5])+ ;
023      B5-> ( "AND" BF1 )+ ;
024       BF1-> "NOT" F1 , F1 [B6] ;
025      B6-> OPREL F1;
026       OPREL-> "=" , "#" , ">=" , ">" , "<" , "<=" ;
027   ID=> "I" ;
028   CTE=> "C" ;
029  $
```

** TABLA DE SIMBOLOS **

N. CLASES

1	X1	2	ID	3	X2	4	X21	5	X3A
6	X4i	7	X4	8	IFC	9	EX	10	F1
11	B5	12	B4	13	X3A1	14	CTE	15	X31
16	X6	17	X5	18	B6	19	ADOP	20	MULOP
21	F2	22	EXBOO	23	EXARI	24	BF1	25	OPREL

S. BASICOS

1	+	2	-	3	*	4	/	5	^
6	(	7	)	8	:=	9	I	10	C
11	ELSE	12	NOT	13	\	14	MOD	15	IF
16	THEN	17	OR	18	AND	19	=	20	#
21	>=	22	>	23	<	24	<=		

FUNCIONES

LA GRAMATICA CONSTA DE 25 NOMBRES DE CLASE, 24 SIMBOLOS BASICOS Y 0 F

```
 1  PMMC,L,A,S,"ANEX4":X1
 2          ROUTIN X1
 3          GOSUB  ID
 4          GOSUB  X2
 5          RETURN
 6          ROUTIN X2
 7          CHECK  8
 8          GOSUB  X21
 9          RETURN
10          ROUTIN X21
11          DECIDE (10,9,6),.F.,@1
12          GOSUB  X3A
13          GOTO   @6
14  @1      DECIDE 2,.F.,@2
15          CHECK  2
16          GOSUB  X41
17          DECIDE (2,1),.F.,@6
18          GOSUB  X4
19          GOTO   @6
20  @2      DECIDE 15,.F.,@3
21          GOSUB  IFC
22          GOSUB  EX
23          CHECK  11
24          GOSUB  EX
25          GOTO   @6
26  @3      DECIDE 12,.F.,@5
27          CHECK  12
28          GOSUB  F1
29          DECIDE 18,.F.,@4
30          GOSUB  B5
31  @4      DECIDE 17,.F.,@6
32          GOSUB  B4
33          GOTO   @6
34  @5      FAULT  260
35  @6      RETURN
36          ROUTIN X3A
37          DECIDE 9,.F.,@9
38          GOSUB  ID
39          DECIDE 8,.F.,@7
40          GOSUB  X2
41          GOTO   @14
42  @7      DECIDE (24,23,22,21,20,19,18,17,14,13,5,4,3,2,1,7),.F.,@8
43          GOSUB  X3A1
44          GOTO   @14
45  @8      FAULT  261
46          GOTO   @14
47  @9      DECIDE (6,10),.F.,@13
48          DECIDE 10,.F.,@10
49          GOSUB  CTE
50          GOTO   @12
51  @10     DECIDE 6,.F.,@11
52          GOSUB  X31
53          GOTO   @12
54  @11     FAULT  517
55  @12     GOSUB  X3A1
56          GOTO   @14
```

```
 57  @13    FAULT  773
 58  @14    RETURN
 59         ROUTIN X3A1
 60         DECIDE 5,.F.,@15
 61         GOSUB  X6
 62  @15    DECIDE (14,13,4,3),.F.,@16
 63         GOSUB  X5
 64  @16    DECIDE (2,1),.F.,@17
 65         GOSUB  X4
 66  @17    DECIDE (24,23,22,21,20,19),.F.,@18
 67         GOSUB  B6
 68  @18    DECIDE 18,.F.,@19
 69         GOSUB  B5
 70  @19    DECIDE 17,.F.,@20
 71         GOSUB  B4
 72  @20    RETURN
 73         ROUTIN X31
 74         CHECK  6
 75         GOSUB  X21
 76         CHECK  7
 77         RETURN
 78         ROUTIN X4
 79  @21    GOSUB  ADOP
 80         GOSUB  X41
 81         DECIDE (2,1),.T.,@21
 82         RETURN
 83         ROUTIN ADOP
 84         DECIDE 1,.F.,@22
 85         CHECK  1
 86         GOTO   @24
 87  @22    DECIDE 2,.F.,@23
 88         CHECK  2
 89         GOTO   @24
 90  @23    FAULT  275
 91  @24    RETURN
 92         ROUTIN X41
 93         GOSUB  F1
 94         DECIDE (14,13,4,3),.F.,@25
 95         GOSUB  X5
 96  @25    RETURN
 97         ROUTIN X5
 98  @26    GOSUB  MULOP
 99         GOSUB  F1
100         DECIDE (14,13,4,3),.T.,@26
101         RETURN
102         ROUTIN MULOP
103         DECIDE 3,.F.,@27
104         CHECK  3
105         GOTO   @31
106  @27    DECIDE 4,.F.,@28
107         CHECK  4
108         GOTO   @31
109  @28    DECIDE 13,.F.,@29
110         CHECK  13
111         GOTO   @31
112  @29    DECIDE 14,.F.,@30
```

```
113           CHECK   14
114           GOTO    @31
115   @30     FAULT   276
116   @31     RETURN
117           ROUTIN F1
118           GOSUB   F2
119           DECIDE 5,.F.,@32
120           GOSUB   X6
121   @32     RETURN
122           ROUTIN F2
123           DECIDE 9,.F.,@33
124           GOSUB   ID
125           GOTO    @36
126   @33     DECIDE 10,.F.,@34
127           GOSUB   CTE
128           GOTO    @36
129   @34     DECIDE 6,.F.,@35
130           GOSUB   X31
131           GOTO    @36
132   @35     FAULT   277
133   @36     RETURN
134           ROUTIN X6
135   @37     CHECK   5
136           GOSUB   F2
137           DECIDE 5,.T.,@37
138           RETURN
139           ROUTIN IFC
140           CHECK   15
141           GOSUB   EXBOO
142           CHECK   16
143           RETURN
144           ROUTIN EX
145           GOSUB   F2
146           GOSUB   X3A1
147           RETURN
148           ROUTIN EXARI
149           GOSUB   F2
150           DECIDE 5,.F.,@38
151           GOSUB   X6
152   @38     DECIDE (14,13,4,3),.F.,@39
153           GOSUB   X5
154   @39     DECIDE (2,1),.F.,@40
155           GOSUB   X4
156   @40     RETURN
157           ROUTIN EXBOO
158           GOSUB   EXARI
159           DECIDE (24,23,22,21,20,19),.F.,@42
160           GOSUB   B6
161           DECIDE 18,.F.,@41
162           GOSUB   B5
163   @41     DECIDE 17,.F.,@45
164           GOSUB   B4
165           GOTO    @45
166   @42     DECIDE 18,.F.,@43
167           GOSUB   B5
168           DECIDE 17,.F.,@45
```

```
169          GOSUB  B4
170          GOTO   @45
171  @43     DECIDE 17,.F.,@44
172          GOSUB  B4
173          GOTO   @45
174  @44     FAULT  270
175  @45     RETURN
176          ROUTIN B4
177  @46     CHECK  17
178          GOSUB  BF1
179          DECIDE 18,.F.,@47
180          GOSUB  B5
181          DECIDE 17,.T.,@46
182          RETURN
183          ROUTIN B5
184  @47     CHECK  18
185          GOSUB  BF1
186          DECIDE 18,.T.,@47
187          RETURN
188          ROUTIN BF1
189          DECIDE 12,.F.,@48
190          CHECK  12
191          GOSUB  F1
192          GOTO   @50
193  @48     DECIDE (10,9,6),.F.,@49
194          GOSUB  F1
195          DECIDE (24,23,22,21,20,19),.F.,@50
196          GOSUB  B6
197          GOTO   @50
198  @49     FAULT  280
199  @50     RETURN
200          ROUTIN B6
201          GOSUB  OPREL
202          GOSUB  F1
203          RETURN
204          ROUTIN OPREL
205          DECIDE 19,.F.,@51
206          CHECK  19
207          GOTO   @57
208  @51     DECIDE 20,.F.,@52
209          CHECK  20
210          GOTO   @57
211  @52     DECIDE 21,.F.,@53
212          CHECK  21
213          GOTO   @57
214  @53     DECIDE 22,.F.,@54
215          CHECK  22
216          GOTO   @57
217  @54     DECIDE 23,.F.,@55
218          CHECK  23
219          GOTO   @57
220  @55     DECIDE 24,.F.,@56
221          CHECK  24
222          GOTO   @57
223  @56     FAULT  281
224  @57     RETURN
```

```
225          ROUTIN ID
226          CHECK  9
227          RETURN
228          ROUTIN CTE
229          CHECK  10
230          RETURN
231          END
 JOB ANEXP4    OFF AT 13:44:03.96  ON 24 AUG  1978
EXECUTION TIME:  00:00:31.27
```

EJEMPLO, CON TRADUCCION A LENGUAJE ENSAMBLADOR.

ANEXO D : ANALIZADOR PARA UNA VARIANTE DEL ALGOL – 60.

JOB ALGRAH ON AT 13:38:20.59 ON 24 AUG 1978

1. GENERACION DEL ANALIZADOR DE ALGOL,
   SIN TRADUCCION A ENSAMBLADOR.

```
MCMP,L,T,M,A,"ANALG"

002   * GRAMATICA ALGOL MODIFICADA
003   *
004   PRODUCCIONES:
005   *
006   PROGRAMA-> BLOSC>
007    BLOSC-> "BEGIN" [DECLARACION ";"]+ (."; " .[SENTENCIA].) "END" ;
008   *
009   * DECLARACIONES
010   *
011    DECLARACION-> DTIPR, DLABEL, DSWITCH ;
012     DTIPR->"OWN" TIPOS VARS / TIPOS (VARS / PROC) / ["RECURSIVE"] PROC
013      TIPOS-> "REAL" / "INTEGER" / "BOOLEAN" ;
014      VARS-> LIDENTIF / "ARRAY" (.",". CONJUNTO .) ;
015      CONJUNTO-> "IDENT" "[" (."," LINF ":" LSUP .) "]" ;
016       LINF-> EXARIT ;
017       LSUP-> EXARIT ;
018     DLABEL-> "LABEL" (. "," . "IDENT" .) ;
019     DSWITCH-> "SWITCH" "IDENT" ":=" (. "," . "IDENT" .) ;
020   *
021   * DECLARACION DE PROCEDIMIENTO
022   *
023    PROC-> "PROCEDURE" PROAB PRPROGR ;
024     PRAB-> "IDENT" [ "(" LIDENTIF ")" ] ";" ["VALUE" LIDENTIF ";"]
025              [ ESPARAM LIDENTIF ";" ]+ ;
026     ESPARAM-> TIPOS [ "ARRAY" / "PROCEDURE" ] / "LABEL" / "SWITCH" /
027                "PROCEDURE" ;
028     LIDENTIF-> (.",". "IDENT" .) ;
029     PRPROGR-> SENTENCIA / "CODE" ;
030   *
031   * SENTENCIAS
032   *
033    SENTENCIA-> [ "ETIDENT" ":" ]+ ( SINCOND, SCOND, SREPET ) ;
034     SINCOND-> SBASICA, BLOSC ;
035     SBASICA-> SASIGNACION, SGOTO, SCASO, SPROCED ;
036      SASIGNACION-> (VARIABLE, "NPRODEF") X2 ;
037      SGOTO-> "GO TO" ( "ETIDENT", "SWIDENT" "[" EXARIT "]" ) ;
038      SCASO-> "CASE" EXARIT "BEGIN" (. ";" . [SENTENCIA] .) "END" ;
039      SPROCED-> "PRIDENT" [ PARACT ] ;
040       PARACT-> "(" (. "," . EXPRESION .) ")" ;
041     SCOND-> IFC SINCOND ["ELSE" SENTENCIA] ;
042      IFC-> "IF" EXBOOL "THEN" ;
043     SREPET-> SFOR, SDO, SWHILE ;
044      SFOR-> "FOR" VARIABLE ":=" LISFOR "DO" SENTENCIA ;
045       LISFOR-> (. "," . EXARIT ["STEP" EXARIT "UNTIL" EXARIT /
046                                 "WHILE" EXBOOL ] .) ;
047      SDO-> "DO" SENTENCIA "UNTIL" EXBOOL ;
048      SWHILE-> "WHILE" EXBOOL "DO" SENTENCIA ;
049   *
050   * EXPRESIONES
051   *
052    X2=> ":=" EXPRES ;
053     EXPRES=> X3A, "-" X41 [X4] , IFC EX "ELSE" EX, "NOT" F1 [B5] [B4]
054      X3A=> VARIABLE ( X2, X3A1), (NUMBS, X31) X3A1 ;
055       X3A1=> [X6] [X5] [X4] [B6] [B5] [B4] ;
056       X31=> "(" EXPRES ")" ;
057        X4=> (ADOP X41 )+ ;
058         ADOP-> "+", "-" ;
059         X41=> F1 [X5];
```

```
060       X5=> ( MULOP F1 )+ ;
061        MULOP-> "*" , "/" , "\" , "MOD" ;
062        F1=> PRIMARIO [X6] ;
063        PRIMARIO=> NUMSS , VARIABLE , FINARIT , X21 ;
064       X6=> ( "^" PRIMARIO )+ ;
065        EX-> PRIMARIO X3A1 ;
066        EXARIT-> PRIMARIO [X6] [X5] [X4] ;
067        EXBOOL-> EXARIT ( B6 [B5] [B4] , B5 [B4] , E4) ;
068      B4-> ( "OR" BF1 [B5])+ ;
069      B5-> ( "AND" BF1 )+ ;
070       BF1-> "NOT" F1 , F1 [B6] ;
071      B6-> OPREL F1;
072       OPREL-> "=" , "#" , ">=" , ">" , "<" , "<=" ;
073   VARIABLE=> "IDENT" [ "[" ( . "," . EXARIT .) "]" ] ;
074   NUMSS=) "CTE" ;
075  *
076   VLOGICO=> "TRUE" , "FALSE" ;
077  *
078     FINARIT-> ("ABS", "SIG", "SQRT", "SIN", "COS", "ARCTAN", "TANH", "
079                  "EXP", "ENTIER", "ROTATE", "TAN") FARACT ;
080  $
```

** TABLA DE SIMBOLOS **

N. CLASES

1	PROGR	2	BLOSC	3	DECLA	4	SENTE	5	DTIPR
6	DLABE	7	DSUIT	8	TIPOS	9	\nRS	10	PROC
11	LIDEN	12	CONJU	13	LINF	14	LSUP	15	EXARI
16	PROAB	17	PRPRO	18	ESPAR	19	SINCO	20	SCOND
21	SREPE	22	SBASI	23	SASIG	24	SGOTO	25	SCASO
26	SPROC	27	VARIA	28	X2	29	PARAC	30	EXPRE
31	IFC	32	EXBOO	33	SFOR	34	SDO	35	SWHIL
36	LISFO	37	X3A	38	X41	39	X4	40	EX
41	F1	42	B5	43	B4	44	X3A1	45	NUMSS
46	X31	47	X6	48	X5	49	B6	50	ADOP
51	MULOP	52	PRIMA	53	FINAR	54	BF1	55	OPREL
56	VLOGI								

S. BASICOS

1	BEGIN	2	;	3	END	4	OWN	5	RECUR
6	REAL	7	INTEG	8	BOOLE	9	ARRAY	10	,
11	IDENT	12	[	13	:	14	]	15	LABEL
16	SWITC	17	:=	18	PROCE	19	(	20	)
21	VALUE	22	CODE	23	ETIDE	24	NPROD	25	GO TO
26	SWIDE	27	CASE	28	PRIDE	29	ELSE	30	IF
31	THEN	32	FOR	33	DO	34	STEP	35	UNTIL
36	WHILE	37	-	38	NOT	39	+	40	*
41	/	42	\	43	MOD	44	^	45	OR
46	AND	47	=	48	#	49	>=	50	>
51	<	52	<=	53	CTE	54	TRUE	55	FALSE
56	ABS	57	SIG	58	SQRT	59	SIN	60	COS
61	ARCTA	62	TANH	63	LN	64	EXP	65	ENTIE
66	ROTAT	67	TAN						

FUNCIONES

LA GRAMATICA CONSTA DE 56 NOMBRES DE CLASE, 67 SIMBOLOS BASICOS Y 0 F

JOB ALGRAM OFF AT 13:39:47.73 ON 24 AUG 1978
EXECUTION TIME: 00:00:49.49

JOB ALGRAN ON AT 13:32:44.48 ON 24 AUG 1970

2. GENERACION DEL ANALIZADOR DE ALGOL,
CON TRADUCCION A ENSAMBLADOR, OBTENIDO
EN UN FICHERO.

```
HCMP,L,T,M,A,"ANALG"

002  * GRAMATICA ALGOL MODIFICADA
003  *
004  PRODUCCIONES:
005  *
006  PROGRAMA-> BLOSC;
007   BLOSC-> "BEGIN" [DECLARACION ";"]+ (."END" .[SENTENCIA].) "END" ;
008  *
009  * DECLARACIONES
010  *
011   DECLARACION-> DTIPR, DLABEL, DSWITCH ;
012    DTIPR->"OWN" TIPOS VARS / TIPOS (VARS / PROC) / ["RECURSIVE"] PROC
013     TIPOS-> "REAL" / "INTEGER" / "BOOLEAN" ;
014     VARS-> LIDENTIF / "ARRAY" (.",". CONJUNTO .) ;
015     CONJUNTO-> "IDENT" "[" (.",". LINF ":" LSUP .) "]" ;
016      LINF-> EXARIT ;
017      LSUP-> EXARIT ;
018    DLABEL-> "LABEL" (. "," . "IDENT" .) ;
019    DSWITCH-> "SWITCH" "IDENT" ":=" (. "," . "IDENT" .) ;
020  *
021  * DECLARACION DE PROCEDIMIENTO
022  *
023   PROC-> "PROCEDURE" PROAB PRPROGR ;
024    PROAB-> "IDENT" [ "(" LIDENTIF ")" ] ";" ["VALUE" LIDENTIF ";"]
025              [ ESPARAM LIDENTIF ";" ]+ ;
026    ESPARAM-> TIPOS [ "ARRAY" / "PROCEDURE" ] / "LABEL" / "SWITCH" /
027                "PROCEDURE" ;
028    LIDENTIF-> (.",". "IDENT" .) ;
029    PRPROGR-> SENTENCIA / "CODE" ;
030  *
031  * SENTENCIAS
032  *
033   SENTENCIA-> [ "ETIDENT" ":" ]+ ( SINCOND, SCOND, SREPET ) ;
034    SINCOND-> SBASICA, BLOSC ;
035    SBASICA-> SASIGNACION, SGOTO, SCASO, SPROCED ;
036     SASIGNACION->  (VARIABLE, "NPRODEF")  X2  ;
037     SGOTO-> "GO TO" ( "ETIDENT", "SLIDENT" "[" EXARIT "]" ) ;
038     SCASO-> "CASE" EXARIT "BEGIN" (. ";" . [SENTENCIA] .) "END" ;
039     SPROCED-> "PRIDENT" [ PARACT ] ;
040      PARACT-> "(" (. "," . EXPRESION .) ")" ;
041    SCOND-> IFC SINCOND ["ELSE" SENTENCIA] ;
042     IFC-> "IF" EXBOOL "THEN" ;
043    SREPET-> SFOR, SDO, SWHILE ;
044     SFOR-> "FOR" VARIABLE ":=" LISFOR "DO" SENTENCIA ;
045      LISFOR-> (. "," . EXARIT ["STEP" EXARIT "UNTIL" EXARIT /
046                                "WHILE" EXBOOL ] .) ;
047     SDO-> "DO" SENTENCIA "UNTIL" EXBOOL ;
048     SWHILE-> "WHILE" EXBOOL "DO" SENTENCIA ;
049  *
050  * EXPRESIONES
051  *
052   X2=> ":=" EXPRES ;
053    EXPRES=> X3A, "-" X41 [X4] , IFC EX "ELSE" EX, "NOT" F1 [B5] [B4]
054     X3A=> VARIABLE ( X2, X3A1), (NUMSS, X31)  X3A1  ;
055      X3A1=> [X6] [X5] [X4] [B6] [B5] [B4] ;
056      X31=> "(" EXPRES ")" ;
057       X4=> (ADOP X41 )+ ;
058        ADOP-> "+", "-" ;
059        X41=> F1 [X5];
```

```
060         X5=> ( MULOP F1 )+ ;
061          MULOP-> "*" , "/" , "\" , "MOD" ;
062          F1=> PRIMARIO [X6] ;
063          PRIMARIO=> NUMSS , VARIABLE , FINARIT , X31 ;
064         X6=> ( "^" PRIMARIO )+ ;
065          EX-> PRIMARIO X3A1 ;
066          EXARIT-> PRIMARIO [X6] [X5] [X4] ;
067          EXBOOL-> EXARIT ( B6 [B5] [B4] , B5 [B4] , B4) ;
068        B4-> ( "OR" BF1 [B5])+ ;
069        B5-> ( "AND" BF1 )+ ;
070         BF1-> "NOT" F1 , F1 [B6] ;
071        B6-> OPREL F1;
072         OPREL-> "=" , "#" , ">=" , ">" , "<" , "<=" ;
073    VARIABLE=> "IDENT" [ "[" (. "," . EXARIT .) "]" ] ;
074    NUMSS=> "CTE" ;
075   *
076    VLOGICO=> "TRUE" , "FALSE" ;
077   *
078      FINARIT-> ("ABS", "SIG", "SQRT", "SIN", "COS", "ARCTAN", "TANH", "
079                  "EXP", "ENTIER", "ROTATE", "TAN") PARACT ;
080   $
```

** TABLA DE SIMBOLOS **

N. CLASES

1	PROGR	2	BLOSC	3	DECLA	4	SENTE	5	DTIPR
6	DLABE	7	DSUIT	8	TIPOS	9	VARS	10	PROC
11	LIDEN	12	CONJU	13	LINF	14	LSUP	15	EXARI
16	PRCAB	17	PRPRO	18	ESPAR	19	SINCO	20	SCOND
21	SREPE	22	SBASI	23	SASIG	24	SGOTO	25	SCASO
26	SPROC	27	VARIA	28	X2	29	PARAC	30	EXPRE
31	IFC	32	EXBOO	33	SFOR	34	SDO	35	SWHIL
36	LISFO	37	X3A	38	X41	39	X4	40	EX
41	F1	42	B5	43	B4	44	X3A1	45	NUMSS
46	X31	47	X6	48	X5	49	B6	50	ADOP
51	MULOP	52	PRIMA	53	FINAR	54	BF1	55	OPREL
56	VLOGI								

S. BASICOS

1	BEGIN	2	;	3	END	4	OUN	5	RECUR
6	REAL	7	INTEG	8	BOOLE	9	ARRAY	10	,
11	IDENT	12	[	13	:	14	]	15	LABEL
16	SWITC	17	:=	18	PROCE	19	(	20	)
21	VALUE	22	CODE	23	ETIDE	24	MPROD	25	GO TO
26	SWIDE	27	CASE	28	PRIDE	29	ELSE	30	IF
31	THEN	32	FOR	33	DO	34	STEP	35	UNTIL
36	WHILE	37	-	38	NOT	39	+	40	*
41	/	42	\	43	MOD	44	^	45	OR
46	AND	47	=	48	#	49	>=	50	>
51	<	52	<=	53	CTE	54	TRUE	55	FALSE
56	ABS	57	SIG	58	SQRT	59	SIN	60	COS
61	ARCTA	62	TANH	63	LN	64	EXP	65	ENTIE
66	ROTAT	67	TAN						

FUNCIONES

LA GRAMATICA CONSTA DE 56 NOMBRES DE CLASE, 67 SIMBOLOS BASICOS Y 0 F

```
  1  PMMC,L,A,"ANALG">PROGR
  2         ROUTIN PROGR
  3         GOSUB  BLOSC
  4         RETURN
  5         ROUTIN BLOSC
  6         CHECK  1
  7  @1     DECIDE (18,16,15,8,7,6,5,4),.F.,@3
  8         GOSUB  DECLA
  9         CHECK  2
 10         GOTO   @1
 11         GOTO   @3
 12  @2     CHECK  2
 13  @3     DECIDE (36,33,32,30,28,27,25,24,23,11,1),.F.,@4
 14         GOSUB  SENTE
 15  @4     DECIDE 2,.T.,@2
 16         CHECK  3
 17         RETURN
 18         ROUTIN DECLA
 19         DECIDE (18,8,7,6,5,4),.F.,@5
 20         GOSUB  DTIPR
 21         GOTO   @8
 22  @5     DECIDE 15,.F.,@6
 23         GOSUB  DLABE
 24         GOTO   @8
 25  @6     DECIDE 16,.F.,@7
 26         GOSUB  DSWIT
 27         GOTO   @8
 28  @7     FAULT  259
 29  @8     RETURN
 30         ROUTIN DTIPR
 31         DECIDE 4,.F.,@9
 32         CHECK  4
 33         GOSUB  TIPOS
 34         GOSUB  VARS
 35         GOTO   @15
 36  @9     DECIDE (8,7,6),.F.,@12
 37         GOSUB  TIPOS
 38         DECIDE (11,9),.F.,@10
 39         GOSUB  VARS
 40         GOTO   @15
 41  @10    DECIDE 18,.F.,@11
 42         GOSUB  PROC
 43         GOTO   @15
 44  @11    FAULT  261
 45         GOTO   @15
 46  @12    DECIDE (18,5),.F.,@14
 47         DECIDE 5,.F.,@13
 48         CHECK  5
 49  @13    GOSUB  PROC
 50         GOTO   @15
 51  @14    FAULT  517
 52  @15    RETURN
 53         ROUTIN TIPOS
 54         DECIDE 6,.F.,@16
 55         CHECK  6
 56         GOTO   @19
```

```
 57  @16    DECIDE 7,.F.,@17
 58         CHECK  7
 59         GOTO   @19
 60  @17    DECIDE 8,.F.,@18
 61         CHECK  8
 62         GOTO   @19
 63  @18    FAULT  264
 64  @19    RETURN
 65         ROUTIN VARS
 66         DECIDE 11,.F.,@20
 67         GOSUB  LIDEN
 68         GOTO   @24
 69  @20    DECIDE 9,.F.,@23
 70         CHECK  9
 71         GOTO   @22
 72  @21    CHECK  10
 73  @22    GOSUB  CONJU
 74         DECIDE 10,.T.,@21
 75         GOTO   @24
 76  @23    FAULT  265
 77  @24    RETURN
 78         ROUTIN CONJU
 79         CHECK  11
 80         CHECK  12
 81         GOTO   @26
 82  @25    CHECK  10
 83  @26    GOSUB  LINF
 84         CHECK  13
 85         GOSUB  LSUP
 86         DECIDE 10,.T.,@25
 87         CHECK  14
 88         RETURN
 89         ROUTIN LINF
 90         GOSUB  EXARI
 91         RETURN
 92         ROUTIN LSUP
 93         GOSUB  EXARI
 94         RETURN
 95         ROUTIN DLABE
 96         CHECK  15
 97         GOTO   @28
 98  @27    CHECK  10
 99  @28    CHECK  11
100         DECIDE 10,.T.,@27
101         RETURN
102         ROUTIN DSUIT
103         CHECK  16
104         CHECK  11
105         CHECK  17
106         GOTO   @30
107  @29    CHECK  10
108  @30    CHECK  11
109         DECIDE 10,.T.,@29
110         RETURN
111         ROUTIN PROC
112         CHECK  18
```

```
113           GOSUB  PRCAB
114           GOSUB  PRPRO
115           RETURN
116           ROUTIN PRCAB
117           CHECK  11
118           DECIDE 19,.F.,@31
119           CHECK  19
120           GOSUB  LIDEN
121           CHECK  20
122   @31     CHECK  2
123           DECIDE 21,.F.,@32
124           CHECK  21
125           GOSUB  LIDEN
126           CHECK  2
127   @32     DECIDE (18,16,15,8,7,6),.F.,@33
128           GOSUB  ESPAR
129           GOSUB  LIDEN
130           CHECK  2
131           GOTO   @32
132   @33     RETURN
133           ROUTIN ESPAR
134           DECIDE (8,7,6),.F.,@35
135           GOSUB  TIPOS
136           DECIDE (18,9),.F.,@39
137           DECIDE 9,.F.,@34
138           CHECK  9
139           GOTO   @39
140   @34     CHECK  18
141           GOTO   @39
142   @35     DECIDE 18,.F.,@36
143           CHECK  15
144           GOTO   @39
145   @36     DECIDE 15,.F.,@37
146           CHECK  16
147           GOTO   @39
148   @37     DECIDE 16,.F.,@38
149           CHECK  18
150           GOTO   @39
151   @38     FAULT  274
152   @39     RETURN
153           ROUTIN LIDEN
154           GOTO   @41
155   @40     CHECK  10
156   @41     CHECK  11
157           DECIDE 10,.T.,@40
158           RETURN
159           ROUTIN PRPRO
160           DECIDE (36,33,32,30,28,27,25,24,23,11,1),.F.,@42
161           GOSUB  SENTE
162           GOTO   @44
163   @42     DECIDE 22,.F.,@43
164           CHECK  22
165           GOTO   @44
166   @43     FAULT  273
167   @44     RETURN
168           ROUTIN SENTE
```

```
169  @45   DECIDE 23,.F.,@46
170        CHECK  23
171        CHECK  13
172        GOTO   @45
173  @46   DECIDE (28,27,25,24,11,1),.F.,@47
174        GOSUB  SINCO
175        GOTO   @50
176  @47   DECIDE 30,.F.,@48
177        GOSUB  SCOND
178        GOTO   @50
179  @48   DECIDE (36,33,32),.F.,@49
180        GOSUB  SREPE
181        GOTO   @50
182  @49   FAULT  260
183  @50   RETURN
184        ROUTIN SINCO
185        DECIDE (28,27,25,24,11),.F.,@51
186        GOSUB  SBASI
187        GOTO   @53
188  @51   DECIDE 1,.F.,@52
189        GOSUB  BLOSC
190        GOTO   @53
191  @52   FAULT  275
192  @53   RETURN
193        ROUTIN SBASI
194        DECIDE (24,11),.F.,@54
195        GOSUB  SASIG
196        GOTO   @58
197  @54   DECIDE 25,.F.,@55
198        GOSUB  SGOTO
199        GOTO   @58
200  @55   DECIDE 27,.F.,@56
201        GOSUB  SCASO
202        GOTO   @58
203  @56   DECIDE 28,.F.,@57
204        GOSUB  SPROC
205        GOTO   @58
206  @57   FAULT  278
207  @58   RETURN
208        ROUTIN SASIG
209        DECIDE 11,.F.,@59
210        GOSUB  VARIA
211        GOTO   @61
212  @59   DECIDE 24,.F.,@60
213        CHECK  24
214        GOTO   @61
215  @60   FAULT  279
216  @61   GOSUB  X2
217        RETURN
218        ROUTIN SGOTO
219        CHECK  25
220        DECIDE 23,.F.,@62
221        CHECK  23
222        GOTO   @64
223  @62   DECIDE 26,.F.,@63
224        CHECK  26
```

```
225         CHECK  12
226         GOSUB  EXARI
227         CHECK  14
228         GOTO   @64
229  @63    FAULT  280
230  @64    RETURN
231         ROUTIN SCASO
232         CHECK  27
233         GOSUB  EXARI
234         CHECK  1
235         GOTO   @66
236  @65    CHECK  2
237  @66    DECIDE (36,33,32,30,28,27,25,24,23,11,1),.F.,@67
238         GOSUB  SENTE
239  @67    DECIDE 2,.T.,@65
240         CHECK  3
241         RETURN
242         ROUTIN SPROC
243         CHECK  28
244         DECIDE 19,.F.,@68
245         GOSUB  PARAC
246  @68    RETURN
247         ROUTIN PARAC
248         CHECK  19
249         GOTO   @70
250  @69    CHECK  10
251  @70    GOSUB  EXPRE
252         DECIDE 10,.T.,@69
253         CHECK  20
254         RETURN
255         ROUTIN SCOND
256         GOSUB  IFC
257         GOSUB  SINCO
258         DECIDE 29,.F.,@71
259         CHECK  29
260         GOSUB  SENTE
261  @71    RETURN
262         ROUTIN IFC
263         CHECK  30
264         GOSUB  EXBOO
265         CHECK  31
266         RETURN
267         ROUTIN SREPE
268         DECIDE 32,.F.,@72
269         GOSUB  SFOR
270         GOTO   @75
271  @72    DECIDE 33,.F.,@73
272         GOSUB  SDO
273         GOTO   @75
274  @73    DECIDE 36,.F.,@74
275         GOSUB  SWHIL
276         GOTO   @75
277  @74    FAULT  277
278  @75    RETURN
279         ROUTIN SFOR
280         CHECK  32
```

```
281         GOSUB   VARIA
282         CHECK   17
283         GOSUB   LISFO
284         CHECK   33
285         GOSUB   SENTE
286         RETURN
287         ROUTIN LISFO
288         GOTO    @77
289   @76   CHECK   10
290   @77   GOSUB   EXARI
291         DECIDE (36,34),.F.,@79
292         DECIDE 34,.F.,@78
293         CHECK   34
294         GOSUB   EXARI
295         CHECK   35
296         GOSUB   EXARI
297         GOTO    @79
298   @78   CHECK   36
299         GOSUB   EXBOO
300   @79   DECIDE 10,.T.,@76
301         RETURN
302         ROUTIN SDO
303         CHECK   33
304         GOSUB   SENTE
305         CHECK   35
306         GOSUB   EXBOO
307         RETURN
308         ROUTIN SWHIL
309         CHECK   36
310         GOSUB   EXBOO
311         CHECK   33
312         GOSUB   SENTE
313         RETURN
314         ROUTIN X2
315         CHECK   17
316         GOSUB   EXPRE
317         RETURN
318         ROUTIN EXPRE
319         DECIDE (53,19,11),.F.,@80
320         GOSUB   X3A
321         GOTO    @85
322   @80   DECIDE 37,.F.,@81
323         CHECK   37
324         GOSUB   X41
325         DECIDE (39,37),.F.,@85
326         GOSUB   X4
327         GOTO    @85
328   @81   DECIDE 30,.F.,@82
329         GOSUB   IFC
330         GOSUB   EX
331         CHECK   29
332         GOSUB   EX
333         GOTO    @85
334   @82   DECIDE 38,.F.,@84
335         CHECK   38
336         GOSUB   F1
```

```
337         DECIDE 46,.F.,@83
338         GOSUB  B5
339  @83    DECIDE 45,.F.,@85
340         GOSUB  B4
341         GOTO   @85
342  @84    FAULT  286
343  @85    RETURN
344         ROUTIN X3A
345         DECIDE 11,.F.,@88
346         GOSUB  VARIA
347         DECIDE 17,.F.,@86
348         GOSUB  X2
349         GOTO   @93
350  @86    DECIDE (52,51,50,49,48,47,46,45,44,43,42,41,40,39,37,35,29,20,
351  ,2),.F.,@87
352         GOSUB  X3A1
353         GOTO   @93
354  @87    FAULT  293
355         GOTO   @93
356  @88    DECIDE (19,53),.F.,@92
357         DECIDE 53,.F.,@89
358         GOSUB  NUMSS
359         GOTO   @91
360  @89    DECIDE 19,.F.,@90
361         GOSUB  X31
362         GOTO   @91
363  @90    FAULT  549
364  @91    GOSUB  X3A1
365         GOTO   @93
366  @92    FAULT  805
367  @93    RETURN
368         ROUTIN X3A1
369         DECIDE 44,.F.,@94
370         GOSUB  X6
371  @94    DECIDE (43,42,41,40),.F.,@95
372         GOSUB  X5
373  @95    DECIDE (39,37),.F.,@96
374         GOSUB  X4
375  @96    DECIDE (52,51,50,49,48,47),.F.,@97
376         GOSUB  B6
377  @97    DECIDE 46,.F.,@98
378         GOSUB  B5
379  @98    DECIDE 45,.F.,@99
380         GOSUB  B4
381  @99    RETURN
382         ROUTIN X31
383         CHECK  19
384         GOSUB  EXPRE
385         CHECK  20
386         RETURN
387         ROUTIN X4
388  @100   GOSUB  ADOP
389         GOSUB  X41
390         DECIDE (39,37),.T.,@100
391         RETURN
392         ROUTIN ADOP
```

```
393          DECIDE 39,.F.,@101
394          CHECK  39
395          GOTO   @103
396  @101    DECIDE 37,.F.,@102
397          CHECK  37
398          GOTO   @103
399  @102    FAULT  306
400  @103    RETURN
401          ROUTIN X41
402          GOSUB  F1
403          DECIDE (43,42,41,40),.F.,@104
404          GOSUB  X5
405  @104    RETURN
406          ROUTIN X5
407  @105    GOSUB  MULOP
408          GOSUB  F1
409          DECIDE (43,42,41,40),.T.,@105
410          RETURN
411          ROUTIN MULOP
412          DECIDE 40,.F.,@106
413          CHECK  40
414          GOTO   @110
415  @106    DECIDE 41,.F.,@107
416          CHECK  41
417          GOTO   @110
418  @107    DECIDE 42,.F.,@108
419          CHECK  42
420          GOTO   @110
421  @108    DECIDE 43,.F.,@109
422          CHECK  43
423          GOTO   @110
424  @109    FAULT  307
425  @110    RETURN
426          ROUTIN F1
427          GOSUB  PRIMA
428          DECIDE 44,.F.,@111
429          GOSUB  X6
430  @111    RETURN
431          ROUTIN PRIMA
432          DECIDE 53,.F.,@112
433          GOSUB  NUMSS
434          GOTO   @116
435  @112    DECIDE 11,.F.,@113
436          GOSUB  VARIA
437          GOTO   @116
438  @113    DECIDE (67,66,65,64,63,62,61,60,59,58,57,56),.F.,@114
439          GOSUB  FINAR
440          GOTO   @116
441  @114    DECIDE 19,.F.,@115
442          GOSUB  X31
443          GOTO   @116
444  @115    FAULT  308
445  @116    RETURN
446          ROUTIN X6
447  @117    CHECK  44
448          GOSUB  PRIMA
```

```
449           DECIDE 44,.T.,@117
450           RETURN
451           ROUTIN EX
452           GOSUB  PRIMA
453           GOSUB  X3A1
454           RETURN
455           ROUTIN EXARI
456           GOSUB  PRIMA
457           DECIDE 44,.F.,@118
458           GOSUB  X6
459   @118    DECIDE (43,42,41,40),.F.,@119
460           GOSUB  X5
461   @119    DECIDE (39,37),.F.,@120
462           GOSUB  X4
463   @120    RETURN
464           ROUTIN EXBOO
465           GOSUB  EXARI
466           DECIDE (52,51,50,49,48,47),.F.,@122
467           GOSUB  B6
468           DECIDE 46,.F.,@121
469           GOSUB  B5
470   @121    DECIDE 45,.F.,@125
471           GOSUB  B4
472           GOTO   @125
473   @122    DECIDE 46,.F.,@123
474           GOSUB  B5
475           DECIDE 45,.F.,@125
476           GOSUB  B4
477           GOTO   @125
478   @123    DECIDE 45,.F.,@124
479           GOSUB  B4
480           GOTO   @125
481   @124    FAULT  288
482   @125    RETURN
483           ROUTIN B4
484   @126    CHECK  45
485           GOSUB  BF1
486           DECIDE 46,.F.,@127
487           GOSUB  B5
488           DECIDE 45,.T.,@126
489           RETURN
490           ROUTIN B5
491   @127    CHECK  46
492           GOSUB  BF1
493           DECIDE 46,.T.,@127
494           RETURN
495           ROUTIN BF1
496           DECIDE 38,.F.,@128
497           CHECK  38
498           GOSUB  F1
499           GOTO   @130
500   @128    DECIDE (67,66,65,64,63,62,61,60,59,58,57,56,53,19,11),.F.,@129
501           GOSUB  F1
502           DECIDE (52,51,50,49,48,47),.F.,@130
503           GOSUB  B6
504           GOTO   @130
```

```
05  @129  FAULT  310
06  @130  RETURN
07        ROUTIN B6
08        GOSUB  OPREL
09        GOSUB  F1
10        RETURN
11        ROUTIN OPREL
12        DECIDE 47,.F.,@131
13        CHECK  47
14        GOTO   @137
15  @131  DECIDE 48,.F.,@132
16        CHECK  48
17        GOTO   @137
18  @132  DECIDE 49,.F.,@133
19        CHECK  49
20        GOTO   @137
21  @133  DECIDE 50,.F.,@134
22        CHECK  50
23        GOTO   @137
24  @134  DECIDE 51,.F.,@135
25        CHECK  51
26        GOTO   @137
27  @135  DECIDE 52,.F.,@136
28        CHECK  52
29        GOTO   @137
30  @136  FAULT  311
31  @137  RETURN
32        ROUTIN VARIA
33        CHECK  11
34        DECIDE 12,.F.,@140
35        CHECK  12
36        GOTO   @139
37  @138  CHECK  10
38  @139  GOSUB  EXARI
39        DECIDE 10,.T.,@138
40        CHECK  14
41  @140  RETURN
42        ROUTIN NUMSS
43        CHECK  53
44        RETURN
45        ROUTIN VLOGI
46        DECIDE 54,.F.,@141
47        CHECK  54
48        GOTO   @143
49  @141  DECIDE 55,.F.,@142
50        CHECK  55
51        GOTO   @143
52  @142  FAULT  312
53  @143  RETURN
54        ROUTIN FINAR
55        DECIDE 56,.F.,@144
56        CHECK  56
57        GOTO   @156
58  @144  DECIDE 57,.F.,@145
59        CHECK  57
60        GOTO   @156
```

```
561  @145  DECIDE 58,.F.,@146
562        CHECK  58
563        GOTO   @156
564  @146  DECIDE 59,.F.,@147
565        CHECK  59
566        GOTO   @156
567  @147  DECIDE 60,.F.,@148
568        CHECK  60
569        GOTO   @156
570  @148  DECIDE 61,.F.,@149
571        CHECK  61
572        GOTO   @156
573  @149  DECIDE 62,.F.,@150
574        CHECK  62
575        GOTO   @156
576  @150  DECIDE 63,.F.,@151
577        CHECK  63
578        GOTO   @156
579  @151  DECIDE 64,.F.,@152
580        CHECK  64
581        GOTO   @156
582  @152  DECIDE 65,.F.,@153
583        CHECK  65
584        GOTO   @156
585  @153  DECIDE 66,.F.,@154
586        CHECK  66
587        GOTO   @156
588  @154  DECIDE 67,.F.,@155
589        CHECK  67
590        GOTO   @156
591  @155  FAULT  309
592  @156  GOSUB  PARAC
593        RETURN
594        END
 ****** T=00006 IS ON LU 62
```

```
 JOB ALGRAM   OFF AT 13:35:11.91  ON 24 AUG  1978
EXECUTION TIME:  00:01:28.37
```

ANEXO E : AUTO - PROCESO DE LOS ANALIZADORES DEL LENGUAJE DE GENERACION Y DE LAS REGLAS EN NOTACION SRL.

1. AUTOCOMPILACION DEL ANALIZADOR DEL PROCESADOR DE MACROS PMMC.

```
 1  PMMC,L,A,B,S,"MENAS":PROG
 2          ROUTIN PROG
 3          CALL   MMC00
 4          CALL   MMC01
 5  L1      GOSUB  LINEA
 6          DECIDE (3,21,22,23,24,25,26,27,28,29),.T,L1
 7          CHECK  2
 8          CALL   MMC03
 9          RETURN
10  *
11          ROUTIN LINEA
12          DECIDE 3,.F,L2
13          CHECK  3
14  L2      GOSUB  INSTR
15          CALL   MMC02
16          CHECK  17
17          RETURN
18  *
19          ROUTIN INSTR
20          DECIDE (21,22,25,27,28),.F.,L7
21          DECIDE 21,.F,L3
22          CHECK  21
23          GOTO   L6
24  L3      DECIDE 22,.F,L4
25          CHECK  22
26          GOTO   L6
27  L4      DECIDE 25,.F,L5
28          CHECK  25
29          GOSUB  LISTA
30          CHECK  16
31          CHECK  6
32          CHECK  16
33          GOTO   L6
34  L5      DECIDE 27,.F.,L51
35          CHECK  27
36          GOTO   L6
37  L51     CHECK  28
38  L6      CHECK  3
39          GOTO   L12
40  L7      DECIDE 23,.F,L8
41          CHECK  23
42          GOTO   L12
43  L8      DECIDE 24,.F,L9
44          CHECK  24
45          CHECK  4
46          GOTO   L12
47  L9      DECIDE 26,.F.,L11
48          CHECK  26
49          DECIDE 4,.F,L10
50          CHECK  4
51          GOTO   L12
52  L10     CHECK  5
53          GOTO   L12
54  L11     CHECK  29
55  L12     RETURN
56  *
```

```
57          ROUTIN LISTA
58          DECIDE 4,.F,L13
59          CHECK  4
60          CALL   MMC16
61          GOTO   L16
62    L13   CHECK  14
63    L14   CHECK  4
64          CALL   MMC16
65          DECIDE 16,.F,L15
66          CHECK  16
67          GOTO   L14
68    L15   CHECK  15
69    L16   RETURN
70    *
71          END
```

```
0001                    ASMB,L
**  NO ERRORS PASS#1 **RTE ASMB 760924**
```

ENSAMBLAJE DEL ANALIZADOR DEL PROCESADOR

DE MACROS PMMC.

```
0001                    RSMB,L
0002   00000                    NAM MENAS,7
0003                            ENT MENAS
0004                            EXT .ENTR
0005   00000 000000  MENAS NOP
0006   00001 016001X            JSB .ENTR
0007   00002 000000R            DEF *-2
0008   00003 016002X            JSB GOSUB
0009   00004 000006R            DEF *+2
0010   00005 000007R            DEF PROG
0011   00006 126000R            JMP MENAS,I
0012                            EXT GOSUB,RETUR,CHECK,DECID,FAULT
0013*
0014****
0015*
0016   00007            PROG    EQU *
0017   00007 016007X            JSB MMC00
0018   00010 000011R            DEF *+1
0019                            EXT MMC00
0020   00011 016010X            JSB MMC01
0021   00012 000013R            DEF *+1
0022                            EXT MMC01
0023   00013 016002X L1         JSB GOSUB
0024   00014 000016R            DEF *+2
0025   00015 000041R            DEF LINEA
0026   00016 016005X            JSB DECID
0027   00017 000023R            DEF *+4
0028   00020 000024R            DEF *+4
0029   00021 000001             DEC 1
0030   00022 000013R            DEF L1
0031   00023 026032R            JMP *+007B
0032   00024 001425             OCT 001425
0033   00025 013027             OCT 013027
0034   00026 014031             OCT 014031
0035   00027 015033             OCT 015033
0036   00030 016035             OCT 016035
0037   00031 000000             OCT 000000
0038   00032 016004X            JSB CHECK
0039   00033 000035R            DEF *+2
0040   00034 000002             DEC   2
0041   00035 016011X            JSB MMC03
0042   00036 000037R            DEF *+1
0043                            EXT MMC03
0044   00037 016003X            JSB RETUR
0045   00040 000041R            DEF *+1
0046****
0047   00041            LINEA   EQU *
0048   00041 016005X            JSB DECID
0049   00042 000046R            DEF *+4
0050   00043 000047R            DEF *+4
0051   00044 000000             DEC 0
0052   00045 000053R            DEF L2
0053   00046 026050R            JMP *+002B
0054   00047 001400             OCT 001400
0055   00050 016004X            JSB CHECK
0056   00051 000053R            DEF *+2
```

```
0057   00052 000003          DEC   3
0058   00053 016002X L2      JSB GOSUB
0059   00054 000056R         DEF *+2
0060   00055 000065R         DEF INSTR
0061   00056 016012X         JSB MMC02
0062   00057 000060R         DEF *+1
0063                         EXT MMC02
0064   00060 016004X         JSB CHECK
0065   00061 000063R         DEF *+2
0066   00062 000021          DEC  17
0067   00063 016003X         JSB RETUR
0068   00064 000065R         DEF *+1
0069****
0070   00065           INSTR EQU *
0071   00065 016005X         JSB DECID
0072   00066 000072R         DEF *+4
0073   00067 000073R         DEF *+4
0074   00070 000000          DEC 0
0075   00071 000175R         DEF L7
0076   00072 026076R         JMP *+004B
0077   00073 012426          OCT 012426
0078   00074 014433          OCT 014433
0079   00075 016000          OCT 016000
0080   00076 016005X         JSB DECID
0081   00077 000103R         DEF *+4
0082   00100 000104R         DEF *+4
0083   00101 000000          DEC 0
0084   00102 000111R         DEF L3
0085   00103 026105R         JMP *+002B
0086   00104 012400          OCT 012400
0087   00105 016004X         JSB CHECK
0088   00106 000110R         DEF *+2
0089   00107 000025          DEC  21
0090   00110 026171R         JMP L6
0091   00111 016005X L3      JSB DECID
0092   00112 000116R         DEF *+4
0093   00113 000117R         DEF *+4
0094   00114 000000          DEC 0
0095   00115 000124R         DEF L4
0096   00116 026120R         JMP *+002B
0097   00117 013000          OCT 013000
0098   00120 016004X         JSB CHECK
0099   00121 000123R         DEF *+2
0100   00122 000026          DEC  22
0101   00123 026171R         JMP L6
0102   00124 016005X L4      JSB DECID
0103   00125 000131R         DEF *+4
0104   00126 000132R         DEF *+4
0105   00127 000000          DEC 0
0106   00130 000153R         DEF L5
0107   00131 026133R         JMP *+002B
0108   00132 014400          OCT 014400
0109   00133 016004X         JSB CHECK
0110   00134 000136R         DEF *+2
0111   00135 000031          DEC  25
0112   00136 016002X         JSB GOSUB
```

```
0113   00137 000141R          DEF *+2
0114   00140 000264R          DEF LISTA
0115   00141 016004X          JSB CHECK
0116   00142 000144R          DEF *+2
0117   00143 000020           DEC  16
0118   00144 016004X          JSB CHECK
0119   00145 000147R          DEF *+2
0120   00146 000006           DEC   6
0121   00147 016004X          JSB CHECK
0122   00150 000152R          DEF *+2
0123   00151 000020           DEC  16
0124   00152 026171R          JMP L6
0125   00153 016005X L5       JSB DECID
0126   00154 000160R          DEF *+4
0127   00155 000161R          DEF *+4
0128   00156 000000           DEC 0
0129   00157 000166R          DEF L51
0130   00160 026162R          JMP *+002B
0131   00161 015400           OCT 015400
0132   00162 016004X          JSB CHECK
0133   00163 000165R          DEF *+2
0134   00164 000033           DEC  27
0135   00165 026171R          JMP L6
0136   00166 016004X L51      JSB CHECK
0137   00167 000171R          DEF *+2
0138   00170 000034           DEC  28
0139   00171 016004X L6       JSB CHECK
0140   00172 000174R          DEF *+2
0141   00173 000003           DEC   3
0142   00174 026262R          JMP L12
0143   00175 016005X L7       JSB DECID
0144   00176 000202R          DEF *+4
0145   00177 000203R          DEF *+4
0146   00200 000000           DEC 0
0147   00201 000210R          DEF L8
0148   00202 026204R          JMP *+002B
0149   00203 013400           OCT 013400
0150   00204 016004X          JSB CHECK
0151   00205 000207R          DEF *+2
0152   00206 000027           DEC  23
0153   00207 026262R          JMP L12
0154   00210 016005X L8       JSB DECID
0155   00211 000215R          DEF *+4
0156   00212 000216R          DEF *+4
0157   00213 000000           DEC 0
0158   00214 000226R          DEF L9
0159   00215 026217R          JMP *+002B
0160   00216 014000           OCT 014000
0161   00217 016004X          JSB CHECK
0162   00220 000222R          DEF *+2
0163   00221 000030           DEC  24
0164   00222 016004X          JSB CHECK
0165   00223 000225R          DEF *+2
0166   00224 000004           DEC   4
0167   00225 026262R          JMP L12
0168   00226 016005X L9.      JSB DECID
```

```
0169  00227 000233R          DEF *+4
0170  00230 000234R          DEF *+4
0171  00231 000000           DEC 0
0172  00232 000257R          DEF L11
0173  00233 026235R          JMP *+002B
0174  00234 015000           OCT 015000
0175  00235 016004X          JSB CHECK
0176  00236 000240R          DEF *+2
0177  00237 000032           DEC  26
0178  00240 016005X          JSB DECID
0179  00241 000245R          DEF *+4
0180  00242 000246R          DEF *+4
0181  00243 000000           DEC 0
0182  00244 000253R          DEF L10
0183  00245 026247R          JMP *+002B
0184  00246 002000           OCT 002000
0185  00247 016004X          JSB CHECK
0186  00250 000252R          DEF *+2
0187  00251 000004           DEC   4
0188  00252 026262R          JMP L12
0189  00253 016004X L10      JSB CHECK
0190  00254 000256R          DEF *+2
0191  00255 000005           DEC   5
0192  00256 026262R          JMP L12
0193  00257 016004X L11      JSB CHECK
0194  00260 000262R          DEF *+2
0195  00261 000035           DEC  29
0196  00262 016003X L12      JSB RETUR
0197  00263 000264R          DEF *+1
0198****
0199  00264           LISTA  EQU *
0200  00264 016005X          JSB DECID
0201  00265 000271R          DEF *+4
0202  00266 000272R          DEF *+4
0203  00267 000000           DEC 0
0204  00270 000301R          DEF L13
0205  00271 026273R          JMP *+002B
0206  00272 002000           OCT 002000
0207  00273 016004X          JSB CHECK
0208  00274 000276R          DEF *+2
0209  00275 000004           DEC   4
0210  00276 016013X          JSB MMC16
0211  00277 000300R          DEF *+1
0212                         EXT MMC16
0213  00300 026327R          JMP L16
0214  00301 016004X L13      JSB CHECK
0215  00302 000304R          DEF *+2
0216  00303 000016           DEC  14
0217  00304 016004X L14      JSB CHECK
0218  00305 000307R          DEF *+2
0219  00306 000004           DEC   4
0220  00307 016013X          JSB MMC16
0221  00310 000311R          DEF *+1
0222                         EXT MMC16
0223  00311 016005X          JSB DECID
0224  00312 000316R          DEF *+4
```

```
 225   00313 000317R          DEF *+4
 226   00314 000000C          DEC 0
 227   00315 000324R          DEF L15
 228   00316 026320R          JMP *+002B
 229   00317 010000C          OCT 010000
 230   00320 016004X          JSB CHECK
 231   00321 000322R          DEF *+2
 232   00322 000020C          DEC  16
 233   00323 026304R          JMP L14
 234   00324 016004X L15      JSB CHECK
 235   00325 000327R          DEF *+2
 236   00326 000017           DEC  15
 237   00327 016003X L16      JSB RETUR
 238   00330 000331R          DEF *+1
 239****
 240                          END
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

```
 JOB JTPMMC   OFF AT 14:00:50.78  ON 24 AUG  1978
EXECUTION TIME:  00:00:14.02
```

2. AUTOCOMPILACION DEL ANALIZADOR DE REGLAS EN SRL.

```
CMP,L,T,M,A,B,S,"MCASR"

02  VOCABULARIO: "NCLASE","SBASICO","FUNCION","=>",";", ",", "+", "(",")"
03         "[", "]", "." )
04  PRODUCCIONES
05  REGLA=> "NCLASE" 'PR' "=>" DEFINICION 'FR' ";" )
06  *
07  DEFINICION=> (. "," 'AL' . 'PS' UNIDAD+ .) )
08  UNIDAD=>"SBASICO" 'T' ["+" 'BUCL'],
09           "NCLASE" 'N' ["+" 'BUCL'],
10           "FUNCION" 'F',
11           "(" ('PI' DEFINICION ")" 'PD' ["+" 'BUCL'], LISTA ")" ),
12           "[" 'CI' (DEFINICION "]" 'CD' ["+" 'AST'] , LISTA "]" 'CD' ));
13  LISTA=> "."'PL' "SBASICO"'T' ["FUNCION"'F']+ "."'ML' UNIDAD+ "."'FL';
14  * FIN DE LA META-METAGRAMATICA
15  $
```

```
** TABLA DE SIMBOLOS **


N. CLASES

  1  REGLA    2  DEFIN    3  UNIDA    4  LISTA


S. BASICOS

  1  NCLAS    2  SBASI    3  FUNCI    4  =>       5  ;
  6  ,        7  +        8  (        9  )       10  [
 11  ]       12  .


FUNCIONES

  1  PR       2  FR       3  AL       4  PS       5  T
  6  BUCL     7  N        8  F        9  PI      10  PD
 11  CI      12  CD      13  AST     14  PL      15  ML
 16  FL

LA GRAMATICA CONSTA DE    4 NOMBRES DE CLASE,  12 SIMBOLOS BASICOS Y  16 F
```

```
PMMC,L,A,S,B,"MCASR";REGLA
        ROUTIN REGLA
        CHECK  1
        CALL   PR
        CHECK  4
        GOSUB  DEFIN
        CALL   FR
        CHECK  5
        RETURN
        ROUTIN DEFIN
        GOTO   @2
@1      CHECK  6
        CALL   AL
@2      CALL   PS
@3      GOSUB  UNIDA
        DECIDE (10,8,3,2,1),.T.,@3
        DECIDE 6,.T.,@1
        RETURN
        ROUTIN UNIDA
        DECIDE 2,.F.,@4
        CHECK  2
        CALL   T
        DECIDE 7,.F.,@13
        CHECK  7                        ANALIZADOR DE REGLAS EN SRL,
        CALL   BUCL
        GOTO   @13                      EN LENGUAJE DE GENERACION.
@4      DECIDE 1,.F.,@5
        CHECK  1
        CALL   N
        DECIDE 7,.F.,@13
        CHECK  7
        CALL   BUCL
        GOTO   @13
@5      DECIDE 3,.F.,@6
        CHECK  3
        CALL   F
        GOTO   @13
@6      DECIDE 8,.F.,@9
        CHECK  8
        DECIDE (10,8,3,2,1),.F.,@7
        CALL   PI
        GOSUB  DEFIN
        CHECK  9
        CALL   PD
        DECIDE 7,.F.,@13
        CHECK  7
        CALL   BUCL
        GOTO   @13
@7      DECIDE 12,.F.,@8
        GOSUB  LISTA
        CHECK  9
        GOTO   @13
@8      FAULT  259
        GOTO   @13
@9      DECIDE 10,.F.,@12
        CHECK  10
```

```
57          CALL    CI
58          DECIDE (10,8,3,2,1),.F.,@10
59          GOSUB   DEFIN
60          CHECK   11
61          CALL    CD
62          DECIDE 7,.F.,@13
63          CHECK   7
64          CALL    AST
65          GOTO    @13
66  @10     DECIDE 12,.F.,@11
67          GOSUB   LISTA
68          CHECK   11
69          CALL    CD
70          GOTO    @13
71  @11     FAULT   515
72          GOTO    @13
73  @12     FAULT   771
74  @13     RETURN
75          ROUTIN LISTA
76          CHECK   12
77          CALL    PL
78          CHECK   2
79          CALL    T
80  @14     DECIDE 3,.F.,@15
81          CHECK   3
82          CALL    F
83          GOTO    @14
84  @15     CHECK   12
85          CALL    ML
86  @16     GOSUB   UNIDA
87          DECIDE (10,8,3,2,1),.T.,@16
88          CHECK   12
89          CALL    FL
90          RETURN
91          END
```

```
0001                    ASMB,L
**  NO ERRORS PASS#1 **RTE ASMB 760924**
```

ENSAMBLAJE DEL ANALIZADOR DE REGLAS EN SRL.

```
0001                    ASMB,L
0002   00000                    NAM MCASR,7
0003                            ENT MCASR
0004                            EXT .ENTR
0005   00000 000000  MCASR NOP
0006   00001 016001X            JSB .ENTR
0007   00002 000000R            DEF *-2
0008   00003 016002X            JSB GOSUB
0009   00004 000006R            DEF *+2
0010   00005 000007R            DEF REGLA
0011   00006 126000R            JMP MCASR,I
0012                            EXT GOSUB,RETUR,CHECK,DECID,FAULT
0013*
0014****
0015*
0016   00007            REGLA EQU *
0017   00007 016004X            JSB CHECK
0018   00010 000012R            DEF *+2
0019   00011 000001             DEC   1
0020   00012 016007X            JSB PR
0021   00013 000014R            DEF *+1
0022                            EXT PR
0023   00014 016004X            JSB CHECK
0024   00015 000017R            DEF *+2
0025   00016 000004             DEC   4
0026   00017 016002X            JSB GOSUB
0027   00020 000022R            DEF *+2
0028   00021 000031R            DEF DEFIN
0029   00022 016010X            JSB FR
0030   00023 000024R            DEF *+1
0031                            EXT FR
0032   00024 016004X            JSB CHECK
0033   00025 000027R            DEF *+2
0034   00026 000005             DEC   5
0035   00027 016003X            JSB RETUR
0036   00030 000031R            DEF *+1
0037****
0038   00031            DEFIN EQU *
0039   00031 026037R            JMP @2
0040   00032 016004X @1         JSB CHECK
0041   00033 000035R            DEF *+2
0042   00034 000006             DEC   6
0043   00035 016011X            JSB AL
0044   00036 000037R            DEF *+1
0045                            EXT AL
0046   00037 016012X @2         JSB PS
0047   00040 000041R            DEF *+1
0048                            EXT PS
0049   00041 016002X @3         JSB GOSUB
0050   00042 000044R            DEF *+2
0051   00043 000066R            DEF UNIDA
0052   00044 016005X            JSB DECID
0053   00045 000051R            DEF *+4
0054   00046 000052R            DEF *+4
0055   00047 000001             DEC 1
0056   00050 000041R            DEF @3
```

```
0057   00051  026055R          JMP  *+004B
0058   00052  005010           OCT  005010
0059   00053  001402           OCT  001402
0060   00054  000400           OCT  000400
0061   00055  016005X          JSB  DECID
0062   00056  000062R          DEF  *+4
0063   00057  000063R          DEF  *+4
0064   00060  000001           DEC  1
0065   00061  000032R          DEF  @1
0066   00062  026064R          JMP  *+002B
0067   00063  003000           OCT  003000
0068   00064  016003X          JSB  RETUR
0069   00065  000066R          DEF  *+1
0070*****
0071   00066             UNIDA EQU  *
0072   00066  016005X          JSB  DECID
0073   00067  000073R          DEF  *+4
0074   00070  000074R          DEF  *+4
0075   00071  000000           DEC  0
0076   00072  000117R          DEF  @4
0077   00073  026075R          JMP  *+002B
0078   00074  001000           OCT  001000
0079   00075  016004X          JSB  CHECK
0080   00076  000100R          DEF  *+2
0081   00077  000002           DEC    2
0082   00100  016013X          JSB  T
0083   00101  000102R          DEF  *+1
0084                           EXT  T
0085   00102  016005X          JSB  DECID
0086   00103  000107R          DEF  *+4
0087   00104  000110R          DEF  *+4
0088   00105  000000           DEC  0
0089   00106  000362R          DEF  @13
0090   00107  026111R          JMP  *+002B
0091   00110  003400           OCT  003400
0092   00111  016004X          JSB  CHECK
0093   00112  000114R          DEF  *+2
0094   00113  000007           DEC    7
0095   00114  016014X          JSB  BUCL
0096   00115  000116R          DEF  *+1
0097                           EXT  BUCL
0098   00116  026362R          JMP  @13
0099   00117  016005X  @4      JSB  DECID
0100   00120  000124R          DEF  *+4
0101   00121  000125R          DEF  *+4
0102   00122  000000           DEC  0
0103   00123  000150R          DEF  @5
0104   00124  026126R          JMP  *+002B
0105   00125  000400           OCT  000400
0106   00126  016004X          JSB  CHECK
0107   00127  000131R          DEF  *+2
0108   00130  000001           DEC    1
0109   00131  016015X          JSB  N
0110   00132  000133R          DEF  *+1
0111                           EXT  N
0112   00133  016005X          JSB  DECID
```

```
0113  00134  000140R          DEF *+4
0114  00135  000141R          DEF *+4
0115  00136  000000C          DEC 0
0116  00137  000362R          DEF @13
0117  00140  026142R          JMP *+002B
0118  00141  003400C          OCT 003400
0119  00142  016004X          JSB CHECK
0120  00143  000145R          DEF *+2
0121  00144  000007           DEC   7
0122  00145  016014X          JSB BUCL
0123  00146  000147R          DEF *+1
0124                          EXT BUCL
0125  00147  026362R          JMP @13
0126  00150  016005X @5       JSB DECID
0127  00151  000155R          DEF *+4
0128  00152  000156R          DEF *+4
0129  00153  000000C          DEC 0
0130  00154  000165R          DEF @6
0131  00155  026157R          JMP *+002B
0132  00156  001400C          OCT 001400
0133  00157  016004X          JSB CHECK
0134  00160  000162R          DEF *+2
0135  00161  000003           DEC   3
0136  00162  016016X          JSB F
0137  00163  000164R          DEF *+1
0138                          EXT F
0139  00164  026362R          JMP @13
0140  00165  016005X @6       JSB DECID
0141  00166  000172R          DEF *+4
0142  00167  000173R          DEF *+4
0143  00170  000000C          DEC 0
0144  00171  000261R          DEF @9
0145  00172  026174R          JMP *+002B
0146  00173  004000C          OCT 004000
0147  00174  016004X          JSB CHECK
0148  00175  000177R          DEF *+2
0149  00176  000010C          DEC   8
0150  00177  016005X          JSB DECID
0151  00200  000204R          DEF *+4
0152  00201  000205R          DEF *+4
0153  00202  000000C          DEC 0
0154  00203  000237R          DEF @7
0155  00204  026210R          JMP *+004B
0156  00205  005010C          OCT 005010
0157  00206  001402           OCT 001402
0158  00207  000400C          OCT 000400
0159  00210  016017X          JSB PI
0160  00211  000212R          DEF *+1
0161                          EXT PI
0162  00212  016002X          JSB GOSUB
0163  00213  000215R          DEF *+2
0164  00214  000031R          DEF DEFIN
0165  00215  016004X          JSB CHECK
0166  00216  000220R          DEF *+2
0167  00217  000011           DEC   9
0168  00220  016020X          JSB PD
```

```
0169   00221  000222R          DEF *+1
0170                           EXT PD
0171   00222  016005X          JSB DECID
0172   00223  000227R          DEF *+4
0173   00224  000230R          DEF *+4
0174   00225  000000C          DEC 0
0175   00226  000362R          DEF @13
0176   00227  026231R          JMP *+002B
0177   00230  003400C          OCT 003400
0178   00231  016004X          JSB CHECK
0179   00232  000234R          DEF *+2
0180   00233  000007           DEC   7
0181   00234  016014X          JSB BUCL
0182   00235  000236R          DEF *+1
0183                           EXT BUCL
0184   00236  026362R          JMP @13
0185   00237  016005X @7        JSB DECID
0186   00240  000244R          DEF *+4
0187   00241  000245R          DEF *+4
0188   00242  000000C          DEC 0
0189   00243  000255R          DEF @8
0190   00244  026246R          JMP *+002B
0191   00245  006000C          OCT 006000
0192   00246  016002X          JSB GOSUB
0193   00247  000251R          DEF *+2
0194   00250  000364R          DEF LISTA
0195   00251  016004X          JSB CHECK
0196   00252  000254R          DEF *+2
0197   00253  000011           DEC   9
0198   00254  026362R          JMP @13
0199   00255  016006X @8        JSB FAULT
0200   00256  000260R          DEF *+2
0201   00257  000403           DEC     259
0202   00260  026362R          JMP @13
0203   00261  016005X @9        JSB DECID
0204   00262  000266R          DEF *+4
0205   00263  000267R          DEF *+4
0206   00264  000000C          DEC 0
0207   00265  000357R          DEF @12
0208   00266  026270R          JMP *+002B
0209   00267  005000C          OCT 005000
0210   00270  016004X          JSB CHECK
0211   00271  000273R          DEF *+2
0212   00272  000012           DEC  10
0213   00273  016021X          JSB CI
0214   00274  000275R          DEF *+1
0215                           EXT CI
0216   00275  016005X          JSB DECID
0217   00276  000302R          DEF *+4
0218   00277  000302R          DEF *+4
0219   00300  000000C          DEC 0
0220   00301  000333R          DEF @10
0221   00302  026306R          JMP *+004B
0222   00303  005010C          OCT 005010
0223   00304  001402           OCT 001402
0224   00305  000400C          OCT 000400
```

```
0225   00306 016002X          JSB GOSUB
0226   00307 000311R          DEF *+2
0227   00310 000031R          DEF DEFIN
0228   00311 016004X          JSB CHECK
0229   00312 000314R          DEF *+2
0230   00313 000013           DEC  11
0231   00314 016022X          JSB CD
0232   00315 000316R          DEF *+1
0233                          EXT CD
0234   00316 016005X          JSB DECID
0235   00317 000323R          DEF *+4
0236   00320 000324R          DEF *+4
0237   00321 000000           DEC 0
0238   00322 000362R          DEF @13
0239   00323 026325R          JMP *+0028
0240   00324 003400           OCT 003400
0241   00325 016004X          JSB CHECK
0242   00326 000330R          DEF *+2
0243   00327 000007           DEC   7
0244   00330 016023X          JSB AST
0245   00331 000332R          DEF *+1
0246                          EXT AST
0247   00332 026362R          JMP @13
0248   00333 016005X @10      JSB DECID
0249   00334 000340R          DEF *+4
0250   00335 000341R          DEF *+4
0251   00336 000000           DEC 0
0252   00337 000353R          DEF @11
0253   00340 026342R          JMP *+0028
0254   00341 006000           OCT 006000
0255   00342 016002X          JSB GOSUB
0256   00343 000345R          DEF *+2
0257   00344 000364R          DEF LISTA
0258   00345 016004X          JSB CHECK
0259   00346 000350R          DEF *+2
0260   00347 000013           DEC  11
0261   00350 016022X          JSB CD
0262   00351 000352R          DEF *+1
0263                          EXT CD
0264   00352 026362R          JMP @13
0265   00353 016006X @11      JSB FAULT
0266   00354 000356R          DEF *+2
0267   00355 001003           DEC     515
0268   00356 026362R          JMP @13
0269   00357 016006X @12      JSB FAULT
0270   00360 000362R          DEF *+2
0271   00361 001403           DEC     771
0272   00362 016003X @13      JSB RETUR
0273   00363 000364R          DEF *+1
0274****
0275   00364           LISTA  EQU *
0276   00364 016004X          JSB CHECK
0277   00365 000367R          DEF *+2
0278   00366 000014           DEC  12
0279   00367 016024X          JSB PL
0280   00370 000371R          DEF *+1
```

```
0281                                    EXT PL
0282   00371 016004X                    JSB CHECK
0283   00372 000374R                    DEF *+2
0284   00373 000002                     DEC   2
0285   00374 016013X                    JSB T
0286   00375 000376R                    DEF *+1
0287                                    EXT T
0288   00376 016005X @14                JSB DECID
0289   00377 000403R                    DEF *+4
0290   00400 000404R                    DEF *+4
0291   00401 000000                     DEC 0
0292   00402 000413R                    DEF @15
0293   00403 026405R                    JMP *+002B
0294   00404 001400                     OCT 001400
0295   00405 016004X                    JSB CHECK
0296   00406 000410R                    DEF *+2
0297   00407 000003                     DEC   3
0298   00410 016016X                    JSB F
0299   00411 000412R                    DEF *+1
0300                                    EXT F
0301   00412 026376R                    JMP @14
0302   00413 016004X @15                JSB CHECK
0303   00414 000416R                    DEF *+2
0304   00415 000014                     DEC  12
0305   00416 016025X                    JSB ML
0306   00417 000420R                    DEF *+1
0307                                    EXT ML
0308   00420 016002X @16                JSB GOSUB
0309   00421 000423R                    DEF *+2
0310   00422 000066R                    DEF UNIDA
0311   00423 016005X                    JSB DECID
0312   00424 000430R                    DEF *+4
0313   00425 000431R                    DEF *+4
0314   00426 000001                     DEC 1
0315   00427 000420R                    DEF @16
0316   00430 026434R                    JMP *+004B
0317   00431 005010                     OCT 005010
0318   00432 001402                     OCT 001402
0319   00433 000400                     OCT 000400
0320   00434 016004X                    JSB CHECK
0321   00435 000437R                    DEF *+2
0322   00436 000014                     DEC  12
0323   00437 016026X                    JSB FL
0324   00440 000441R                    DEF *+1
0325                                    EXT FL
0326   00441 016003X                    JSB RETUR
0327   00442 000443R                    DEF *+1
0328****
0329                                    END
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

JOB XTMCMP OFF AT 13:58:05.87 ON 24 AUG 1978
EXECUTION TIME: 00:00:25.77