TRATAMIENTO UNIFICADO DE LA COMUNICACION Y TRANSFORMACION DE PROCESADORES

M. BERTRAN J. XAMPENY

Se describe una metodología para abordar problemas relacionados con la transformación de textos y su uso en el desarrollo de los programas para el proyecto CONCE, la parte central del futuro sistema de telecontrol computadorizado de la red eléctrica de ENHER.

La metodología se basa en el uso de una versión de TBNF, el TBNF-BASIC (T-BASIC), un lenguaje sencillo adaptado a la traducción o transformación de textos. Sus aplicaciones en el proyecto incluyen entre otras, la definición concisa y la programación de diálogos hombre-máquina y máquina-máquina, la mecanización de aspectos repetitivos de la programación y la construcción de preprocesadores de lenguajes fuente con vistas a su mejor estructuración.

El esfuerzo adicional involucrado en la especificación, diseño y puesta a punto del T-BASIC viene grandemente compensado con la simplificación del desarrollo y mantenimiento, con el consiguiente ahorro de tiempo, que su disponibilidad permite. El empleo de una herramienta adecuada lleva como consecuencia la eliminación de aspectos mecánicos no esenciales en la especificación y programación de los algoritmos para abordar los problemas mencionados.

1. INTRODUCCION

El desarrollo de programas para un sistema de telecontrol centralizado de una red eléctrica constituye un proyecto de considerable complejidad. Ello es debido no tan solo a la gran cantidad y diverso tipo de elementos eléctricos a representar conjuntamente con sus estados, sino a la necesidad de que el sistema conozca la interrelación entre los mismos y entre las diversas funciones de control formando un conjunto único y coherente cuyo estado ha de seguir en tiempo real a fin de dar información oportuna a los operadores de la red eléctrica, frecuentemente en forma de esquemas gráficos.

Las funciones de cálculo para simulación, optimización, etc. han de basarse en un único modelo simplificado del sistema contenido en la base de datos y reflejo constante del estado variante de la red eléctrica. Programas de captación de datos, análisis, validación y simplificación de los mismos, construyen aquél a partir de otro modelo detallado, también contenido en la base de datos, en el que se reflejan tanto los elementos fijos como sus interconexiones y asociaciones físicas, y de los datos llegados de las estaciones remotas de telemetría. Algunos aspectos de esta transformación se detallan en /1/. Los programas de dibujo de esquemas usan también los dos tipos de información juntamente con otro asimismo residente en la base de datos, resultado del procesado de los primeros.

Muchos de los problemas involucrados en el desarrollo pueden reducirse al procesado de textos. Por texto entenderemos una secuencia de elementos o símbolos, no limitándose éstos a los alfanuméricos sino incluyendo los posibles contenidos de los campo de mensajes, por ejemplo. El texto no debe necesariamente residir en un fichero sino que puede estar descompuesto en subunidades cuyo procesado debe hacerse a medida que se genera cada una de ellas.

El análisis y las acciones desencadenadas en un procesador por la llegada de mensajes de comunicaciones constituye un ejemplo de proceso de textos.

Los problemas de interacción hombre-máquina, de comunicación entre procesos (dos programas dentro de un mismo ordenador o en dos di

- M. Bertran d'ENHER, Provença, 386, Barcelona 25, i col.laborador a la ETSETB, Baixa de S. Pere, 7. Barna. 3.
- J. Xampeny d'ENHER, Provença, 386, Barcelona 25.
- Article rebut l'Agost del 1978.

ferentes), de mecanización del desarrollo, - de la generación de una base de datos y de - traducción de lenguajes constituyen también casos particulares de procesado de textos.

Uno de los problemas del procesado será decidir sobre la corrección o incorrección del - texto analizado. Las técnicas de análisis -- sintáctico, ampliamente desarrolladas en el área de los compiladores, serán pues aplicables aquí en un contexto más general.

El método heurístico con que se abordó en un principio el diseño de compiladores fué superado por métodos basados en la descripción - formal de lenguajes /2/. La BNF /3/ es una - notación práctica para expresar las gramáticas de los lenguajes y sus traductores. El - empleo de la forma TBNF /4/, basada grande-- mente en la BNF, permite economizar senten-- cias al definir formalmente un programa traductor para gramáticas insensibles al contexto y en particular regulares. Para lenguajes cuyas gramáticas son sensibles al contexto - existen otros métodos para su definición /5/. Nuestro enfoque aborda tan solo gramáticas - insensibles al contexto, no obstante la parte sensible al contexto podría también tra-- tarse diseñando las acciones apropiadas.

Aplicado al caso concreto de confección de - traductores, el T-BASIC proporcionará una -- forma más elegante y concisa de codificar el traductor de nuestro BASIC-CCUPB, por ejem-- plo; representa pues el siguiente paso lógico que se apunta en /6/. El lenguaje de im-- plantación de sistemas de la universidad de Linköping, SILL /7/, representó una generalización de la TBNF; incluía la posibilidad de definir estructuras de datos tipo PASCAL /8/ y primitivas para sincronizar procesos ade-- más de construcciones tipo TBNF. Otras generalizaciones al área de diálogos hombre-má-- quina y a las comunicaciones se describen en /9/ y /10/ respectivamente. En la segunda -- también se incluyen primitivas para sincronización de procesos. Nuestro enfoque es más - modesto, ya que nos limitamos a programas secuenciales, eliminamos por tanto del T-BASIC la sincronización de procesos. Hemos juntado un mínimo indispensable de primitivas de control de TBNF con un grupo de instrucciones - máquina suficientes para extender el lenguaje del área exclusiva de generación de códigos a la más general del proceso y transformación de textos.

El uso del T-BASIC permite desarrollar fácilmente programas para mecanización del desarrollo: Inserciones automáticas de declaraciones de áreas de datos, parametrización - de las dimensiones de las mismas, editores especializados, preprocesadores, confección paralela y simultánea de la documentación, etc. El área de los sistemas de desarrollo de Software es reciente y se potenciará grandemente en el futuro /11/. Los sistemas eficientes se adaptarán a áreas específicas: - tiempo-real, comercio, simulación, etc. Al no poder disponer de un buen sistema de desarrollo para el proyecto CONCE, suplimos - su falta con la mecanización de todos los - aspectos posibles de la confección de pro-- gramas y con la inclusión de cuantas herramientas de evaluación y análisis de los mismos podamos disponer. El T-BASIC simplifica grandemente dicho proceso.

Con el presente artículo no se pretende una descripción rigurosa del lenguaje ni de sus aplicaciones. Tan solo nos proponemos ilustrar la practicabilidad del método descri-- biendo las ideas principales en que se basa mediante su aplicación a situaciones diferentes dentro de un mismo proyecto. Hemos se-- leccionado para ello ejemplos en diversas - áreas del proyecto CONCE, fijando la aten-- ción tan solo en aquellos de sus aspectos - que más ilustran la utilidad de la herramienta para el procesado de textos. Esperamos - con ello que quede claro el método unificado y el encuadramiento de los problemas, -- más que los detalles de los mismos.

2. PROBLEMAS DE PROCESADO DE TEXTOS

Seguidamente pasamos a describir una serie de problemas que posteriormente trataremos con un enfoque común. Todos ellos giran directa o indirectamente alrededor del mantenimiento de una base de datos de tiempo -- real ya sea a través de acciones de los operadores de la red eléctrica, del análisis - de los mensajes que pueden llegar de una -- red de comunicaciones o de los cambios de - estructura de la red que deben introducirse periódicamente (nuevas líneas, subestacio-- nes, etc...)

2.1 Interacciones hombre-máquina

La generación en un tiempo aceptable de una base de datos para tiempo real que englobe de una forma coherente la información de una red eléctrica, requiere el uso de herramientas adecuadas que faciliten la introducción de datos detectando posibles errores. La definición precisa de un diálogo hombre-máquina debe figurar en el programa constitutivo de dicha herramienta. Las frecuentes modificaciones que se introducirán en la base de datos, una vez el sistema esté operando, vendrán también facilitadas por la disponibilidad del generador de la base de datos.

Otro caso de interacción hombre-máquina se tiene cuando los operadores del sistema eléctrico piden información, ejecutan telemandos o activan los programas de simulación y vigilancia involucrados en el sistema de tiempo real. No es necesario insistir en la necesidad de que este diálogo hombre-máquina esté clara e inambiguamente definido. Con este fin se ha creido conveniente la introducción de una consola inteligente especial cuyo programa analice las secuencias de pulsadores accionados por los operadores, detectando e inactivando las inválidas y formando mensajes para el ordenador principal reflejando compactamente la información contenida en las correctas.

Ambos casos de interacción hombre-máquina pueden englobarse dentro de un proceso de transformación de textos y más específicamente de traducción. Las secuencias de pulsadores accionados por el operador constituyen sentencias del lenguaje fuente que son traducidas a una forma compacta de codificación antes de ser transferidas al resto del sistema. Una explicación más elaborada de este punto puede verse en /9/.

2.2 Comunicación entre procesos

Un caso frecuente en los sistemas de tiempo real es el intercambio de mensajes entre diferentes procesos. Tal situación se tiene, ya sea entre dos programas residentes en un mismo ordenador o bien en dos procesadores diferentes. Normalmente los diálogos constan de secuencias de mensajes en ambas direcciones, existiendo secuencias válidas y secuencias inválidas. El conjunto ordenado de mensajes en una dirección puede tratarse como un texto a analizar por el proceso receptor. Durante el análisis, el proceso receptor emitirá mensajes en la dirección opuesta y ejecutará cambios en su propio estado. El mismo caso se tiene si se mira el proceso emisor, ya que deberá recibir y analizar la secuencia de mensajes a él enviados por el primero.

2.3 Mecanización del desarrollo

Razones de transportabilidad aconsejan que la mayoría de los programas de tiempo real estén codificados en Fortran. Al utilizar distintas tablas de la base de datos, sus listados contendrán la lista de nombres de todos los vectores constitutivos de las mismas, juntamente con sus tamaños. Las listas se repetirán muchas veces en los fuentes de los distintos programas. El hecho de que al modificar el tamaño de un vector, todas las versiones deban modificarse a la vez, aconseja tener una copia única en el sistema para ser intercalada automáticamente en los puntos convenientes de los fuentes Fortran. De esta forma se eliminará una fuente común de errores. Este proceso constituye claramente un ejemplo, aunque sencillo, de un proceso de transformación de textos. Un texto con referencias simbólicas a las diversas tablas se transforma en el texto en el que los símbolos han sido sustituidos por los listados completos.

Por otro lado, los tamaños de los vectores dependen de los parámetros de dimensionalidad del sistema eléctrico. En general existirá una fórmula para el tamaño de cada vector en función de los parámetros. Los cambios de dimensionalidad son frecuentes tanto en la etapa de generación como durante la operación. Nuevamente, con el objeto de eliminar otra fuente de errores, es útil el mantener un fichero con la lista de los vectores de cada tabla figurando las fórmulas simbólicas en vez de los enteros indicadores de sus dimensiones obligatorios en Fortran. Ello requiere disponer de un procesador de fórmulas simbólicas que las traduzca al valor entero correspondiente, generando de este modo un fuente Fortran correcto. El texto simbólico es pues transformado a su equivalente Fortran en función de los parámetros de dimensionalidad del sistema eléc-

trico.

2.4 Documentación

La documentación de los programs es un aspecto imprescindible en un producto de calidad. Debe, no obstante, subrayarse que aquella no debe suplir las deficiencias del diseño. Un programa poco pensado, lleno de "parches" e initeligible necesitará muchas páginas de documentación, por lo general de poco uso. Un buen diseño con conceptos claramente definidos y con el número de opciones reducidas al mínimo imprescindible será de fácil documentación.

Empezar el proceso de documentación antes -- del de codificación contribuye significativamente a clarificar y acelerar el segundo. -- Gran parte de la documentación puede incluirse dentro de los programas fuente en forma - de comentarios. Estos no son meramente aclaraciones o anotaciones al margen, sino que - constituyen por si solos una explicación concisa de lo que el programa debe hacer. Entre las sentencias y las frases de los comentarios se intercalarán los trozos del lenguaje fuente que realizan lo indicado de una forma menos explícita en los primeros.

Los comentarios pueden a su vez distribuirse en dos niveles. Un primer nivel sería explicativo del objeto del módulo que sigue a continuación mientras que el segundo nivel explicaría con términos más precisos cómo se - ejecuta lo indicado en el nivel primero.

El conjunto de comentarios englobados en el primer nivel constituye por si solo una enumeración autoexplicativa de los módulos y -- sus funciones. La totalidad de los comentarios, primero y segundo nivel, da una explicación más detallada de las funciones y de - como son ejecutadas. El tercer nivel de documentación vendría constituido por las sentencias funte del programa.

Naturalmente, no se podrá evitar un nivel cero de documentación escrita que defina la filosofía de conjunto y la interrelación de -- los módulos para constituir la unidad que debe ser el producto.

En el sistema de documentación integrada en el programa fuente que se ha descrito, es - necesario disponer de listadores selectivos. Para tener una visión global del sistema -- bastará con leer un escrito de los comentarios englobados en el nivel uno. En cambio, para un estudio más detallado de un módulo será bueno estudiar un listado de comentarios del primer y segundo nivel conjuntamente. La confección de listados selectivos es, aunque sencilla, una operación de transformación de textos.

2.5 Traducción de lenguajes

La traducción de lenguajes es también un -- ejemplo claro de transformación de textos. Su aplicación en el proyecto de un sistema de tiempo real puede tenerse al desear trabajar con una versión estructurada de Fortran. Para ello se requiere transformar -- construcciones del tipo "if ... then ... else" a sus equivalentes en Fortran.

Aunque no directamente relacionado con el - proyecto, el proceso de traducción de un BASIC fuente a un BASIC intermedio queda englobado en el tipo general de procesado de textos que abordaremos a continuación. La - herramienta que introduciremos proporcionará un medio de programación rápida del traductor; representará por lo tanto la continuación del camino apuntado en /6/.

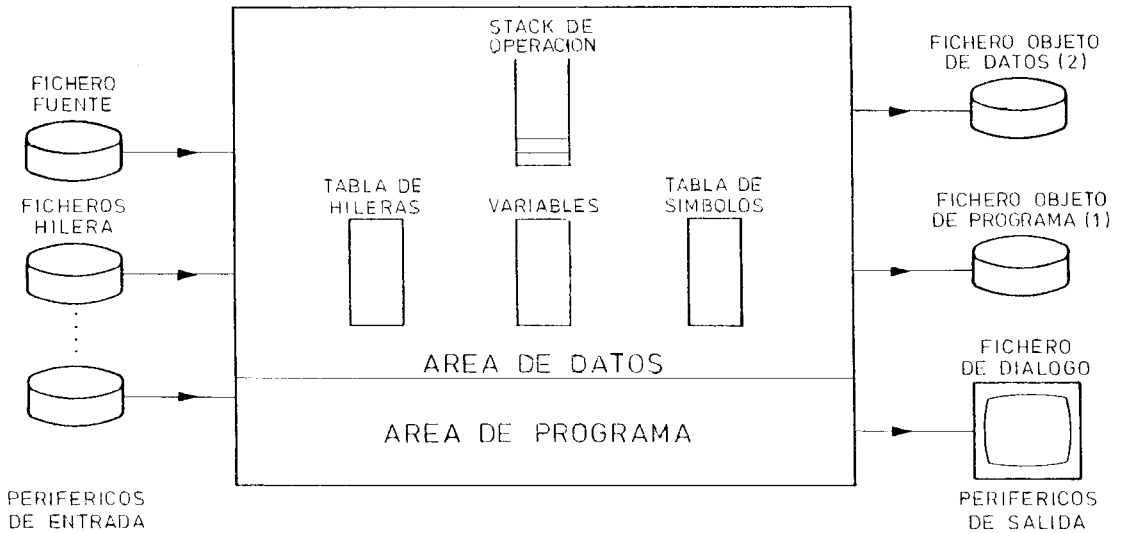
2.6 Análisis del registro histórico

El sistema de tiempo real generará un ficheros histórico de registro de sucesos eléctricos y de sistema, juntamente con estados globales instantáneos de la red eléctrica. A partir de un fichero único deberán generarse ficheros específicos destinados a aplicaciones muy variadas.

La selección y compactación de la información contenida en el fichero original con - la creación de los ficheros específicos -- constituye también un campo de aplicación - de la misma metodología.

3. SOLUCION UNIFICADA

Como se ha visto, los problemas descritos -



MAQUINA IDEAL TRADUCTORA

Fig. 1

en el apartado anterior son casos particulares de un proceso de transformación de textos. Podrían abordarse con un procesador -- ideal de ficheros, uno principal, que denominaremos el fuente, y los demás secundarios que contendrán en general, unidades de texto a insertar en puntos convenientes del -- fuente, les denominaremos ficheros hilera. El procesador dejaría los textos resultado en ficheros objeto. En principio, aunque podrían existir varios de ellos, limitaremos el modelo a dos que denominaremos objeto de datos y objeto de programa. Finalmente será conveniente incluir un fichero nuevo de salida, que denominaremos de diálogo. Los casos de interactividad podrán usar éste fi-- chero. El término fichero se usa aquí en un sentido genérico y puede representar varios tipos de periféricos.

El procesador ideal (véase figura 1) analizará el texto del fichero fuente para dejar textos transformados en los ficheros objeto de datos y objeto de programa. En general - se valdrá del de diálogo para generar mensajes explicativos del estado del procesador o de sus necesidades.

Podemos visualizar el procesador como una - máquina ideal con sus áreas de programa y - de datos y sus periféricos. La inicializa-- ción del procesador consistirá en almacenar en ellas los datos y el programa traductor que debe ejecutarse. Las instrucciones de - la máquina ideal, de que se compone el programa traductor, corresponderán a operacio-- nes típicas de análisis y traducción de textos.

Entre los diferentes constituyentes de los - datos figuran: la tabla de hileras, la de variables y la de símbolos. La primera consta de una lista de mnemotécnicos de las hileras junto con información sobre el fichero en -- que residen o bien, caso de hileras cortas, sobre la localización de la hilera en el á-- rea de datos. La tabla de variables comprende una lista de mnemotécnicos con sus equivalentes binarios. Los mnemotécnicos pueden -- usarse, ya sea como instrucciones para generar su código binario o como variables en -- una fórmula del texto fuente que debe ser -- evaluada. En ambos casos la máquina usará el equivalente binario del mnemotécnico. Finalmente algunas hileras del texto fuente pue-- den guardarse en la tabla de símbolos juntamente con el número total de octetos generados en el objeto en el momento de aparecer - el símbolo. La posición de un símbolo dentro de la tabla puede a su vez añadirse al texto objeto. Ello hace posible traducir programas con subrutinas.

La construcción de un interpretador de la máquina ideal permite abordar los diversos problemas apuntados en la sección anterior con solo cambiar el contenido de las áreas de datos y de programa de la misma.

La implantación del interpretador valiéndose

de un lenguaje de alto nivel contribuye a su transportabilidad.

Programar directamente en el lenguaje de la máquina ideal los diversos traductores y transformadores de textos, no contribuye a la claridad de los programs ni a la rapidez de su desarrollo. Simplifica enormemente el trabajo posterior, programar los diversos procesadores de texto en un lenguaje de alto nivel, TBNF-BASIC ó T-BASIC, y codificar tan solo en lenguaje máquina el traductor de programas escritos en T-BASIC al lenguaje de la máquina ideal.

En realidad el T-BASIC engloba una estructura de control de la ejecución y de operaciones lógicas adaptada al análisis de textos. La estructura sintáctica del texto a analizar queda claramente reflejada en el programa codificado en T-BASIC. Aquella está formada por la combinación de un subconjunto de construcciones de la TBNF. El analizador sintáctico implícito en todo programa T-BASIC es del tipo "top-down" (ver /2/). Consta tan solo de la combinación de cinco construcciones que, para aquellos lectores no versados en TBNF, describimos brevemente en el apéndice. Estas son: Encabezamiento de subrutina, yuxtaposición o bien operación lógica "y", operación lógica "o", negación y secuencia delimitada. Para indicar la búsqueda inmediata de una hilera de caracteres en el texto basta incluirla entre comillas "HILERA 1", por ejemplo.

La ejecución viene determinada por el fichero fuente cuya estructura válida está especificada por el programa. El análisis del texto, empezando por el símbolo de la izquierda procede hacia la derecha. No obstante, en algunos casos es conveniente poder volver atrás. La posición del último símbolo correctamente analizado del texto fuente le llamamos itexte. El programa es un evaluador lógico constante de la conformidad del fuente con la estructura que define el programa. Las transferencias de control están implícitas en el programa y se hacen de acuerdo con la progresión de la evaluación. Así por ejemplo dentro de la construcción

```
( <A> <B> | <C> )
```

la ejecución empieza intentando hallar el grupo <A> seguido del <B>.

Si encuentra el <A> pero no el <B> a continuación, empieza de nuevo intentando hallar el <C>. Si tampoco lo hallara, la máquina quedaría en estado de falsedad después de la ejecución del grupo, y la ejecución procedería según sea la situación del mismo dentro del programa. Caso de haberse satisfecho el grupo <A> <B> o bien el <C>, el estado sería de verdad y la ejecución procedería con la construcción situada a la derecha del grupo.

Como puede intuirse, el lenguaje no se reduce a analizar textos y por lo tanto tan solo a efectuar evaluaciones lógicas y transferencias de control. En ciertos momentos de la ejecución deben llevarse a cabo acciones. Estas equivalen en realidad a instrucciones de la máquina ideal. Entre ellas están las que permiten leer nuevo texto y mueven el puntero de texto, itexte; las que permiten llenar los ficheros objeto con código de diverso tipo, por ejemplo instrucciones, trozos del texto fuente, hileras, etc.; las que permiten realizar operaciones aritméticas; y otras.

Las acciones deben insertarse en los puntos oportunos de la estructura de control. Así pues, si en la estructura anterior se desea que una vez reconocido el grupo <A> seguido del <B> se genere el código de la instrucción LOAD y en caso de haberse reconocido el <C> la STORE, deberá programarse

```
( <A> <B> GENC (LOAD) | <C> GENC (STORE))
```

haciendo uso de la acción GENC para añadir al objeto el código binario de la instrucción cuyo mnemotécnico va entre paréntesis.

Para ver que la herramienta es en efecto útil para abordar los problemas antes descritos, comentaremos a continuación trozos de programas T-BASIC, para tratar algunos de ellos. De esta forma quedará más claro el uso de la estructura de control y el de las diversas acciones.

4. LISTADOR SELECTIVO

Se trata de transferir al fichero objeto programa tan solo las sentencias del fichero fuente que empiecen con "C1" o bien con "C2". Si el fichero objeto es una impresora o TRC se obtendrá un listado de la documentación

de nivel 1 y 2. El ejemplo es sencillo pero nos servirá para ilustrar el estilo de programación y la mecánica de la ejecución. La totalidad del listador se resume en cuatro subrutinas:

```
<LISTSEL> ::= DSEQ 1 - ... (<SENTEN>) ( );
<SENTEN> ::= LREG (<CONC1OC2> | <RESTO>);
<CONC1OC2> ::= ('C1' | 'C2') GNTTX;
FI;
```

La primera nos indica la estructura del fichero fuente compuesto de una secuencia ilimitada de sentencias sin delimitador. La segunda las clasifica, después de leerlas (acción para leer un registro fuente: LREG), en aquellas que empiezan con "C1" o "C2" y las restantes. La tercera después de haber hallado "C1" o "C2" transfiere al fichero objeto toda la tabla de texto (acción GNTTX). Caso contrario, la evaluación de su grupo "o" dejará a la máquina en estado de falsedad con lo que <CONC1OC2> en la segunda subrutina será también falso y el control pasará a evaluar <RESTO>, la segunda opción del grupo "o", que será siempre verdad ya que en ella tan solo se ejecuta la acción NOP equivalente a la operación nula.

5. EVALUADOR DE FORMULAS

Se debe aquí procesar la declaración de un área de datos en la que aparecen fórmulas para especificar las dimensiones de sus vectores o matrices. El resultado debe ser la obtención, en el fichero objeto de programa, de un texto Fortran. Deberemos para ello evaluar las fórmulas. Los valores de las variables que en ellas aparecen corresponderán a variables de la máquina T-BASIC, sus valores estarán por lo tanto contenidos en su área de datos.

Un ejemplo corto de declaración a procesar puede ser del tipo

```
 DIMENSION
*       A(¡NUM+2, ¡LIN+NUM-1), C(¡NUM)
*       D(¡LIN)
```

en la que, como se ve, las fórmulas a evaluar van precedidas de un signo de exclamación. Ello no es estrictamente necesario pero mejora la eficiencia del evaluador. El fichero objeto que debería generarse sería, por ejemplo:

```
 DIMENSION
*       A(22,29), C(20)
*       D(10)
```

en donde se ha supuesto que las variables NUM y LIN toman los valores 20 y 10 respectivamente.

En el caso real se admite todo tipo de fórmulas, vamos, no obstante, a limitarnos a sumar y restar a fin de simplificar la explicación.

El programa evaluador consta de seis subrutinas. La primera es igual a la del listador, por ello tan solo fijaremos nuestra atención en las cinco restantes. En la segunda y tercera

```
<SENTEN> ::= LREG <NOFRMULA>
             DSEQ 0 - ... (<FORMULA>) (<NOFRMULA>)
             <NOFRMULA> ESCR;
<NOFRMULA> ::= DSEQ 1-80 (NO('¡') GNCR) ( );
```

por un lado se describe la estructura de todo registro a procesar: fórmulas encuadradas dentro de hileras de carácteres sin representar fórmula alguna (no-fórmulas). Por otro lado se especifica la sintaxis de una no-fórmula como una secuencia de caracteres sin delimitador distintos del carácter "¡", teniendo cuidado de transferir cada uno de ellos al fichero objeto mediante la acción GNCR.

Las tres últimas subrutinas del programa

```
<FORMULA> ::= '¡' <TERM> DSEQ 0 - ...
              (<MASFORML>) ( ) GNCP 5;
<MASFORML> ::= ('+' <TERM> SUMA
               | '-' <TERM> RESTA);
<TERM> ::= (CVARF | NUM);
FI;
```

tienen por objeto evaluar las fórmulas valiéndose del "stack" de operaciones de la máquina T-BASIC, dejando sus valores, codificados en cinco caracteres decimales (formato I5), en las siguientes posiciones del fichero objeto mediante la acción GNCP 5 que convierte el valor binario situado en la cabeza del "stack". La acción NUM deja en la cabeza del "stack" el número entero que sigue en el texto. La acción CVARF dejará en cambio el valor de la variable cuyo mnemotécnico sigue

en el texto. Ambas, CVARF o NUM, pueden dejar la máquina en estado de falsedad si no encuentran un mnemotécnico o bien un entero en las siguientes posiciones del texto. Finalmente las acciones SUMA y RESTA, de significado obvio, actuan siempre sobre las dos posiciones primeras del "stack" sustituyéndolas por el resultado. El programa evalúa la fórmula en el "stack" a medida que la analiza.

6. INSERTADOR DE DECLARACIONES

Este es el primer ejemplo en el que interviene más de un fichero fuente. Se trata de procesar un fichero fuente principal que contiene sentencias con referencias simbólicas a algún fichero de declaración tal como puede ser el resultado del ejemplo descrito anteriormente. Debe confeccionarse un programa que, reconociendo las referencias, las sustituya en el fichero objeto de programa por las declaraciones equivalentes. Una referencia al fichero de declaración con nombre DATOS empezaría en el primer carácter de la sentencia y sería, por ejemplo, C*DATOS. Así pues los caracteres C* encabezando una sentencia indican referencia simbólica a la hilera cuyo nombre sigue.

Los ficheros fuente de declaraciones son asimilados a ficheros hilera de la máquina T-BASIC, con nombres coincidentes con las referencias simbólicas.

La acción GNFFILFO interpreta el nombre que sigue en el texto como el de una hilera residente en memoria externa. Después de comprobar la existencia de la hilera, ésta es transferida al fichero objeto de programa; caso de que la comprobación resulte negativa, la máquina queda en estado de falsedad.

La primera subrutina sería idéntica a la de los ejemplos anteriores por lo que no la repetimos, las dos restantes son:

```
<SENTEN> ::= LREG ('C*' <NOMDECLA> | GNTTX);
<NOMDECLA> ::= GNFFILFO FSUB1;
FI;
```

la segunda, después de leer un registro fuente lo pasaría al objeto (acción GNTTX), caso de no existir los caracteres "C*" en primera posición. De no ser así se invoca la acción GNFFILFO, dentro ya de la tercera subrutina.

La acción FSUB equivale a NOP siempre que la máquina esté en estado de verdad, caso contrario transfiere control a una subrutina a definir por el usuario en Fortran y cuyo número figura a continuación, en nuestro caso la subrutina 1. Aquí el usuario escribiría algún mensaje de advertencia y la ejecución seguiría, transfiriéndose al objeto (acción GNTTX) la sentencia encabezada por la hilera "C*" y cuya referencia simbólica no ha sido reconocida.

7. DIALOGO ENTRE PROCESOS

Dos programas dentro de un mismo ordenador que se comunican mediante el intercambio de mensajes constituyen un ejemplo de diálogo entre dos procesos. Ello no deja de ser cierto si ambos programas residen en diferentes procesadores. Como se mencionó en la introducción, una consola especial equipada con su microprocesador constituye el terminal de diálogo de los operadores con el sistema de telecontrol, el programa residente en la consola debe analizar las secuencias de pulsaciones accionadas por los operadores, transformando las correctas en mensajes para el ordenador central, y rechazando las secuencias incorrectas.

Un programa en el ordenador central debe estar en espera de los mensajes, analizarlos al recibirlos, efectuar las operaciones implicadas y contestar con otro mensaje el programa de la consola. Muchas operaciones de telecontrol necesitan de varios mensajes con sus correspondientes contestaciones. Existirán pues secuencias de mensajes válidas y secuencias de mensajes inválidas. El lenguaje T-BASIC constituye una buena herramienta para la especificación concisa de los diálogos hombre/máquina y máquina/máquina que aparecen en estos procesos de comunicación. Lo veremos con el ejemplo del diálogo de fijación de umbrales de vigilancia de magnitudes analógicas.

Las opciones disponibles para ésta función son introducir o anular los límites de vigilancia para varias magnitudes eléctricas situadas en distintos puntos de la red. El sis

tema generará alarmas al rebasarse los límites. Veremos el programa de la consola y el del ordenador central.

Por el lado de la consola, el operador deberá primero seleccionar el punto concreto (estación, parque, barras o interruptor de salida de línea), seguidamente deberá especificar de que magnitud se trata y entrar el valor límite o bien el signo de anulación. El diálogo se completa mediante una pantalla (TRC) en la que el ordenador debe visualizar un listado con los límites y valores actuales para las diversas magnitudes. Hay un listado por cada parque y deberá empezarse por pedir dicho listado.

El fichero fuente de la máquina T-BASIC estará asignado a las teclas especiales. El código de cada tecla llenará los octetos del "buffer" del texto fuente. En el objeto de programa se irá formando el mensaje. La acción T (AA) tiene el objeto de reconocer si el código del octeto siguiente en el texto es el de la tecla cuyo mnemotécnico es AA. Es paralela al uso de 'BC', por ejemplo, en el caso de querer hallar los códigos ASCII de una B seguida de una C.

Una primera especificación del diálogo es la siguiente

```
<UMBRALES> ::= <SELISTA> T (ELEM/LIS)
               <NUMERO>(T(I.MAX) | T(V.MAX)
                | T(V.MIN)) (<NUMDECIM>
                | T(ANULAR)) <ENVIO/RS>
                T(ACCION) <ENVIO/RS>;
<SELISTA> ::= T(ESTACION) <NUMERO> DSEQ 0-1
              (<TENSNOMI>) ( ) T(ELEM/LIS)
              <NUMERO> T(LISTA) <ENVIO/RS>;
<TENSNOMI> ::= (T(380KV) | T(220KV)
               | T(110KV));
<NUMERO> ::= NUM;
```

en la que se puede apreciar claramente las secuencias correctas a pulsar. Obsérvese que si no se pulsa ninguna tensión nominal (<TENSNOMI>) la selección de lista es también válida, no obstante se selecciona entonces una lista de estación y no de parque.

A medida que va analizando las teclas, la consola debe formar un mensaje en el "buffer" de programa objeto. El formato del mismo es como sigue:

```
|_ 2|_ _ 5|_ _ 8|_ _11|_ _14|_ _17|_ _ _21|
 tip  est   tens  det   num   func. num. dec
```

Los campos deben llenarse de caracteres ASCII. Una vez formado el mensaje, éste debe transmitirse al ordenador central y la consola debe quedar esperando un mensaje que le de permiso para continuar. El permiso de continuación del ordenador central está formado por los caracteres "PC" seguidos del código de la función en tres caracteres decimales. La llamada a <ENVIO/RS> engloba la transmisión, el análisis del permiso de continuación y la retransmisión en caso de que el permiso no sea válido. La subrutina está codificada con la hipótesis de que el código de la tecla de función está en la cabeza del "stack". En su primera parte se llena el campo de "función" (tres dígitos) y se transmite todo el mensaje (ESCR). Seguidamente se conmuta el fichero fuente de las teclas al ordenador principal (FIFIL (ORDPRINC)), cuyo canal está asignado al fichero hilera ORDPRINC. Después de leer el mensaje (LREG) se analiza comprobando que su número coincida con el código de tecla enviado, caso contrario se llama nuevamente a la secuencia <ENVIO/RS>. La subrutina es como sigue:

```
<ENVIO/RS> ::=              POBJ  14
                            GRVAR (CODTECL)
                            GNCP  3
                                        ESCR
   FIFIL (ORDPRINC)       LREG
 ('PC' NUML 3=CODTECL     |   CVAR (CODTECL)
            <ENVIO/RS>     )
   FIFONT   ;
```

Para completar la descripción del programa de la consola detallamos la subrutina <SELISTA> que reconoce las teclas de selección de lista pulsadas por el operador y envía el mensaje apropiado al ordenador principal. Su codificación es

```
<SELISTA> ::=               POBJ  0
                            GNFIL (PSEUDORD)
            T(ESTACION) <NUMERO>
            DSEQ 0-1 (<TENSNOMI>) ()
            T(ELEM/LIS)     POBJ 11
            <NUMERO>
            T(LISTA)        CVAR (DEMLISTA)
            <ENVIO/RS>     ;
```

en ella, además de la parte sintáctica o de

control lógico ya expuesta, están las acciones para llenar el campo de "tipo de mensaje", caracteres 1 y 2, con la hilera (de dos caracteres) de mnemotécnico PSEUDORD; y para dejar en la cabeza del "stack" el código de la función demanda de lista (CVAR(DEMLISTA)), en preparación de la llamada <ENVIO/RS> ya explicada.

El programa residente en el ordenador principal es una secuencia indefinida de lecturas y análisis de mensajes de operador, éstos podrán ser de tipo selección o pseudo-orden (PO), o bien de órdenes ejecutivas o acciones (OA). Esta estructura queda claramente reflejada en las dos subrutinas primeras del programa.

```
<PROGPRIN> ::= DSEQ 1-...(LREC <MENSOPE>)();
<MENSOPE> ::= ('PO'<PSEUDORD>|'OA'<ORDENACC>);
```

Examinaremos someramente la subrutina <PSEUDORD> ya que <ORDENACC> tiene estructura análoga. Su función es decodificar los números decimales que figuran en los campos de localización: Estación, Tensión, Detalle y número y dejarlos en cuatro posiciones en la cabeza del "stack". Según sea el código del campo función, se pasa control a la acción de usuario correspondiente (ACCU i) que generalmente comporta modificaciones en la pantalla del TRC que indican usualmente al operador la operación que desea efectuar; por otro lado la función oportuna queda en estado de selección. Finalmente la subrutina forma y envía el mensaje de permiso de continuación. El código de tres de las subrutinas puede verse a continuación:

```
<PSEUDORD> ::= <LOCALIZA>
            (TFIL(CODMLIST)          ACCU 1
                                     CVAR (DEMLISTA)
            | TFIL(CODI.MAX) NUML 4  ACCU 2
                                     CVAR (INMX)
                 ...
            | TFIL(CODANVMN)         ACCU 7
                                     CVAR (AVMN)
                          )
            <PERMISO>;
<LOCALIZA> ::= DSEQ 4 - 4  (NUML 3)();
<PERMISO>  ::= GNFL (PC)             GNCP 3
                                          ESCR
                        ;
```

Obsérvese como la acción CVAR deja en la cabeza del "stack" el código de la función para que la subrutina <PERMISO> lo coloque en el lugar apropiado del mensaje de permiso mediante la acción GNCP. La acción ESCR envía el permiso a la consola.

8. GENERADOR INTERACTIVO DE UNA BASE DE DATOS

Un parque es un conjunto de interruptores y embarrados que tiene por objeto unir de distintas maneras varias líneas, transformadores y otros elementos eléctricos convergentes al parque. A un parque le corresponderán dos tablas en la base de datos, la primera contendrá información sobre sus elementos y sus relaciones; la segunda sobre su esquema para poderlo dibujar en el TRC a petición de los operadores.

Desde un teclado alfanumérico, al que asimilaremos el fichero fuente, y trabajando con un TRC de caracteres, direccionable aleatoriamente, se trata de definir un parque de una red eléctrica. Para simplificar supondremos que aquél consiste solamente de tres tipos de elementos: barras, interruptores y salidas de líneas. Mientras se van definiendo los elementos se debe llenar, por un lado un fichero de disco con registros de codificación interna de los elementos y sus interrelaciones, por otro se ha de actualizar la tabla que codifica el dibujo del parque. Esta última se actualizará mediante acciones de usuario oportunas (ACCUi).

A medida que se define cada nuevo elemento, se dibuja en la pantalla el esquema actualizado mediante la invocación a otra acción definida por el usuario (ACCU 3).

Existen dos modos de operación: inicialización y cambios. El primero corresponde a la definición de un nuevo parque, todavía no definido, mientras que el segundo introduce modificaciones sobre la información previamente definida de un parque. Ambos modos generan el mismo tipo de registros, no obstante seleccionamos arbitrariamente el fichero objeto de datos para que el programa lo llene con los registros de inicialización, mientras que en modo de cambios los registros se añadirán al fichero objeto de programa.

Una acción de usuario, ACCU 10, se activa al terminar el procesado de un parque para combinar ambos ficheros sustiuyendo los registros antiguos oportunos por los del fichero de cambios, construyendo de este modo el fichero de la información actualizada de parque. Las primeras cinco subrutinas del generador de parques son las siguientes:

```
<GENPARC> ::= <NUMPARQE>                    ACCU 3
              <INIOCAMB>                    ACCU 10
                          ;
<NUMPARQE> ::= (<BORRAREA>              PREG (NUMYNOMB)
                                  LREG
                     NUM      SIMBL       ACCU2
                   | <NUMPARQE>       ) ;
<INIOCAMB> ::= (<BORRAREA>              PREG (INIOCAMB)
                                  LREG
               ('IN'<INICIAR>
               |'CM'<CAMBIAR>)
               |<INIOCAMB>      ) ;
<BORRAREA> ::=                          CURX 1
                                        CURY 2
                                    PREG (BLANCS)
                                        CURX 1
                                        CURY 2  ;
```

La acción ACCU 2 tiene por objeto actualizar la matriz de esquema y llenar, en su caso, el objeto de programa con los registros ya existentes de codificación interna del parque. Ello lo hace a partir del número y nombre simbólico del parque que las acciones NUM y SIMBL han dejado en cabeza del "stack" y en la primera entrada de la tabla de símbolos. Si dentro de ACCU 2 se detecta alguna incoherencia entre el número y el símbolo la máquina T-BASIC queda en estado de falsedad.

Obsérvese como, caso de error en las entradas, ya sea éste sintáctico o de otro tipo, se activa nuevamente la subrutina <NUMPARQE> que volverá a pedir la misma información al individuo que genera la base de datos. El mismo estilo de programación se puede observar también en la subrutina <INIOCAMB>. Como su nombre indica, la subrutina <BORRAREA>, borra el área del TRC a partir del punto x=1, y=2 o sea las dos últimas líneas, dejando el cursor posicionado en el mismo punto. La acción PREG transfiere a la pantalla la hilera cuyo mnemotécnico sigue entre paréntesis: NUMYNOMB, INIOCAMB y BLANCS para pedir el número y nombre del parque, si se desea trabajar en modo de inicialización o de cambios, o bien para transferir blancos.

Comentaremos brevemente las cuatro subrutinas siguientes solamente.

```
<INICIAR> ::=                           SELEOBJ 2
              <DATOS> ;
<CAMBIAR> ::=                           SELEOBJ 1
              <DATOS> ;
<DATOS> ::= <BARRAS> <INTERRUP> <SALIDLIN> ;
<BARRAS> ::= DSEQ 0 - ... (<BORRAREA> PREG (NUEVBAR?)
                                  LREG
            'SI'<BARRA>                 ESCR
                        ) ( ) ;
```

La acción SELEOBJ selecciona el fichero objeto oportuno según el modo de operación deseado.

La subrutina <DATOS> nos informa de la secuencia en que se entrará la información del parque: barras, interruptores y salidas de líneas. Dentro de la subrutina <BARRAS> se genera un registro objeto (ESCR) después de que la subrutina <BARRA> haya reconocido correctamente la información de cada barra.

9. PREPROCESADOR DE CONSTRUCCIONES CONDICIONALES

A fin de conseguir fuentes más leibles, es conveniente el uso de estructuras IF...THEN ...ELSE, DO...WHILE, etc... disponibles, por ejemplo, en PL/I o bien PASCAL, pero no en FORTRAN. Al querer, por otro lado, mantener los programas en FORTRAN, se deberá construir un traductor que convierta las construcciones al código FORTRAN equivalente. Daremos un ejemplo para la construcción IF. Por descontado siempre queda el recurso de adquirir el preprocesador, no obstante puede programarse con relativa facilidad en T-BASIC.

En este ejemplo, el preprocesador no admitirá una sentencia IF que no sea del tipo

```
IF    <CLAUSLOG>   THEN
        BEGIN
            [ <SENTES> ]
            [   (1)    ]
        END
      ELSE
        BEGIN
            [ <SENTES> ]
            [   (2)    ]
        END
```

en donde <CLAUSLOG> representa una cláusula lógica y <SENTES> es un conjunto de sentencias Fortran que si contiene alguna condicional ha de ser forzosamente del tipo descrito. El código FORTRAN equivalente podría ser el siguiente

```
        IF  (<NOCLAUS>)  GO TO a
                ┌ <SENTES> ┐
                │   (1)    │
                └          ┘
              GO TO b
a       CONTINUE
        ┌ <SENTES> ┐
        │   (2)    │
        └          ┘
b       CONTINUE
```

en donde <NOCLAUS> es la cláusula lógica original negada. Se reservarán los números 9000 en adelante para las etiquetas a y b. El primer IF procesado dará lugar a las etiquetas 9000 y 9001, el segundo a la 9002 y 9003, -- etc...

El programa T-BASIC, correspondiente a las - suposiciones mencionadas, consta de diez subrutinas. Fijaremos nuestra atención solamente en las tres más significativas. Sin considerar, de momento, las acciones; su estructura sintáctica será:

```
<SENTES> ::= DSEQ 1 - ... ((<GRUPOIF>
             | <FORTSENT>)) ();
<GRUPOIF> ::= <BLANCOS> 'IF'
              <CLAUSLOG> 'THEN' <BLOQUE>
              <BLANCOS>  'ELSE' <BLOQUE> ;
<BLOQUE> ::= <BLANCOS> 'BEGIN' <SENTES> 'END';
```

lo que nos ilustra la estructura supuesta -- del texto fuente compuesta de grupos IF in-- tercalados entre sentencias FORTRAN. Obsérve se que dentro de la subrutina <BLOQUE> aparece una llamada a la <SENTES>. Ello no presenta problema alguno ya que las rutinas T-BA-- SIC son recursivas.

Debemos ahora intercalar las acciones en los lugares apropiados a fin de completar el programa T-BASIC. Estas deben por un lado generar las hileras "IF (",") GO TO", "GO TO" y "CONTINUE"; y por otro los enteros correspondientes a a y b. Para lo primero se usará la acción GNFIL (generar hilera), para lo segundo se mantendrá en la variable ETIQUETA el - primer entero disponible (≥ 9000) y se calcularán en el "stack" los dos enteros a y b. - Se usarán las acciones CVAR, GRVAR y SUMCi - para transferir valores entre una variable y el "stack" y para añadirle la constante i. Las demás acciones usadas ya han salido anteriormente.

Las tres producciones quedarán finalmente, - una vez completadas con las acciones, tal como se reproduce a continuación

```
<SENTES> ::= DSEQ 1 - ... (          LREG
                  (<GRUPOIF>|<FORTSENT>)
                        ) ( );
<GRUPOIF> ::= <BLANCOS> 'IF'          GNFIL (IF()
              <CLAUSLOG> 'THEN'       GNFIL(CPGOTO)
                        DSEQ 5-5(CVAR(ETIQUETA))()
                                      SUMC 2
                                      GRVAR(ETIQUETA)
                                      POP
                                      GNCP 5  ESCR
              <BLOQUE>                GNFIL(BLANCS)
                                      GNFIL(GO TO)
                                      SUMC 1
                                      GNCP 5   ESCR
                                    LREG
              <BLANCOS> 'ELSE'        GNCP 5
                                      GNFIL(CONTINUE)
                                           ESCR
              <BLOQUE>                SUMC 1
                                      GNCP 5
                                    GNFIL(CONTINUE)
                                           ESCR

<BLOQUE> ::=                      LREG
              <BLANCOS> 'BEGIN'
              <SENTES>  'END'    ;
```

10. CONCLUSIONES

La exposición de programas T-BASIC para la - resolución de problemas de interacción hombre-máquina, máquina-máquina, de generación interactiva de bases de datos, de traducción de lenguajes y de mecanización del desarrollo y documentación de programas; incluidos dentro del área del procesado y transformación de textos y que aparecen en diversas -- partes del proyecto CONCE, muestra las ventajas de usar una misma herramienta de definición y programación de algoritmos para abordar los diferentes problemas.

El nivel de abstracción de las estructuras - de control incluidas en T-BASIC requeriría - mucho más detalle caso de usar otro lenguaje

de alto nivel para su programación, por lo - tanto la claridad de los algoritmos queda notablemente mejorada, la longitud de los programas acortada y los errores de programación reducidos en número.

Todo ello repercute en la reducción del coste de desarrollo y mantenimiento de los programas de transformación de textos. La aplicación de una misma metodología para la especificación y programación de muchas partes - del mismo proyecto contribuye asimismo a la simplicidad de su implementación y a la reducción de la documentación necesaria.

El T-BASIC tiene un área de aplicación muy - concreta. En él se ha buscado un compromiso entre la amplitud y generalidad de aquella y el coste de desarrollo del lenguaje. Por ello se ha buscado en el T-BASIC la minimización del número de construcciones de control y acciones en él incluidas conjuntamente con la maximización de su área de aplicabilidad. Todo ello hace que el esfuerzo adicional que la especificación, diseño y puesta a punto - del T-BASIC ha requerido quede compensado con las ventajas que su disponibilidad comporta.

11. REFERENCIAS

/1/ BERTRAN, M. "A stack algorithm for power network connectivity determination". Proceedings 20th Midwest Symposium on Circuits and Systems. Lubbock, Texas. Agosto 1977, pp. 428-432.

/2/ AHO, A.V., ULLMAN, J.D. "The theory of - parsing, translation, and compiling". Prentice-Hall, New Jersey, 1972.

/3/ NAUR, P. et al. "Report on the algorithmic language ALGOL 60". Comm. ACM, 3. 1960. pp. 299.

/4/ LAWSON, H.W., DOUCETTE, D.R. "A translation machine with automated Top-Down parsing". ACM Sigplan Notices, v.11, n.2, - 1976.

/5/ GINSBURG, S., ROUNDS, E.M. "Dynamic syntax specification using grammar forms". IEEE Trans. on Software Engineering, v. SE-4, n.1, 1978, pp. 44-55.

/6/ BERTRAN, M., BANQUE, F., PORCAR, J.M. "BASIC CCUPB: Un pas vers la generació - automàtica de traductors". Qüestiió, v.1 n.1, 1977, pp.25-34.

/7/ LAWSON, H.W. "Técnicas recientes de implantación de software de sistemas", ATI Barcelona, Junio 1974.

/8/ WIRTH, N. "The programming language Pascal", Acta Informática, v.1, n.1, 1971, pp. 35-63.

/9/ LAWSON, H.W., BERTRAN, M., SANAGUSTIN, J. "The formal definition of Human/machine communications". Software: Practice - and Experience, v.8, n.1, 1978, pp. 51-58.

/10/ LAWSON, H.W., COHEN, B. "The definition and implementation of telecommunication languages". Proc. Eigth Ann. Symp. on Human Factors in Telecommunications. 1977.

/11/ RAMAMOORTHY, C.V., HO, SIU-BUN F. "Testing large software with automatic software evaluation systems". IEEE Trans. on Software Engineering, v. SE-1, n.1, 1975, pp. 46-58.

12. APENDICE

Construcciones de control de ejecución de TBNF incluidas en T-BASIC.

Definición de un grupo de símbolos.

El simbolismo "<xxx> ::= ... ;" engloba la - definición de la estructura del grupo de símbolos del texto fuente con nombre "xxx". Equivale al encabezamiento y cierre de una - subrutina.

Yuxtaposición de grupos.

El simbolismo "<xxx> <yyy>" a la derecha de un encabezamiento indica que al grupo "yyy" debe figurar a continuación del xxx en el texto. Equivale a la operación lógica "y".

Secuencia delimitada.

Para indicar que una secuencia de grupos <xxx>, entre k y m, separados por grupos <yyy>, debe aparecer en el texto, se usa el simbolismo "DSEQ k-m (<xxx>) (<yyy>)". En realidad, en vez de los grupos <xxx> e <yyy>

podrían figurar estructuras más complejas, - combinación siempre de las básicas.

Grupo "o" lógico.

Para indicar varias posibilidades en un punto del texto se emplea el simbolismo "(a | b | ... | n)" en donde a, b, c, ..., n representan las estructuras posibles.

Negación.

Para especificar la no presencia de una es-- tructura determinada en un punto del texto - se incluye aquélla entre paréntesis precedida de la hilera NO : "NO(...)". Caso de no - existir la estructura se avanza una posición el puntero de texto y la máquina queda en estado de verdad.