ANALISIS SINTACTICO DEL LENGUAJE NATURAL SIN COMPILADORES ESPECIALES

por E. Casabán Moya *

Resumen

El artículo trata del diseño de mecanismos y algoritmos para el proceso de listas en lenguajes corrientes de cálculo numérico. Para hacer esto posible, ha sido necesario traducir a lenguaje de alto nivel los mecanismos normalmente suministrados por los compiladores (representación rectangular de los datos y pilas o stacks*). Como ejemplo de uso de tales mecanismos se presenta un analizador sintáctico de inglés.*

Abstract

The article deals with the design of mechanisms and algorithms for the list processing in current languages of numerical calculation. To make this possible, it was necessary to put into a high-level language the mechanisms usually supplied by the compilers (rectangular representation of data and stacks). As an example of the use of these mechanisms an automatic syntactic analizer of English is presented.

(*) Del Servicio de Documentación e Informática médica. Hospital clínico universitario. Valencia.

El problema que presentamos a continuación lo hemos tomado de Foster (1967). Hemos diseñado para resolverlo un algoritmo, inmediatamente traducible a lenguajes de alto nivel corrientes, que incluye mecanísmos normalmente suministrados por los compiladores de los sistemas orientados a la manipulación de símbolos. Sistemas como IPL, LISP o SNOBOL, por ejemplo.

Tales mecanísmo son los que hacen posible la representación interna de las estructuras de listas, y entre ellas se incluye también la pila o *stack*, necesaria para los procesos recursivos (1).

La función del algoritmo consiste en decidir si una serie de números naturales, que representan distintas "partes de la oración" (nombre, adjetivo, verbo, etc.), constituye una estructura oracional correcta o no de acuerdo con una gramática generativa inglesa que se consulta durante el proceso.

La gramática pertenece a las llamadas *context-free phrase structure grammars* y viene representada por una estructura de listas. El método de análisis sigue el expuesto por Kuno y Oettinger (1963).

La gramática en nuestro caso tan sólo intenta ilustrar un procedimiento. Consta de dos conjuntos de reglas correspondientes a *noun-phrase* y *sentence*, respectivamente.

Los exponemos a continuación.

1º

noun-phrase . . . . . the noun
noun-phrase . . . . . the adjective noun
noun-phrase . . . . . the noun noun-phrase verb
noun-phrase . . . . . the adjective noun noun-phrase verb

2º

Sentence . . . . . the noun is adjective
Sentence . . . . . the adjective noun is adjective
Sentence . . . . . the noun noun-phrase verb is adjective
Sentence . . . . . the adjective noun noun-phrase verb is adjective

Las partes de la oración que pueden aparecer en nuestras oraciones son:

"the "	*Representado por :*	1
noun		2
adjective		3
"is "		4
verb		5

(1) El lector que desconozca los conceptos apuntados por esta terminología puede consultar el libro de Foster (1967).

La estructura de listas correspondientes a *noun-phrase* es:

((THE, NOUN), (THE, ADJECTIVE, NOUN), (THE, NOUN, NOUN-PHRASE, VERB), (THE, ADJECTIVE, NOUN, NOUN-PHRASE, VERB)).

En nuestra codificación:

((1,2), (1,3,2), (1,2,23,5), (1,3,2,23,5))

La estructura de listas correspondiente a *sentence* es:

((THE, NOUN, IS, ADJECTIVE), (THE, ADJECTIVE, NOUN, IS, ADJECTIVE), (THE, NOUN, NOUN-PHRASE, VERB, IS, ADJECTIVE), (THE, ADJECTIVE, NOUN, NOUN-PHRASE, VERB, IS, ADJECTIVE)).

En nuestra codificación:

((1,2,4,3), (1,3,2,4,3), (1,2,23,5,4,3), (1,3,2,23,5,4,3)).

Las dos listas forman una estructura que representa la totalidad de la gramática (fig. 1). En ella, los números menores que 10 son átomos ("partes de la oración") y los mayores que 10 son siempre direcciones (*addresses*).

La estructura de la gramática se almacena en memoria a base de dos *arrays* unidimensionales H y T cuyas variables, al conectarse formalmente por el índice (H(i) y T(i)), conservan cada uno de los rectángulos de la estructura.

El algoritmo empieza por leer el primer término de la oración a analizar. Todas nuestras oraciones es preceptivo que comiencen con el artículo *the*.

Esta palabra se compara con la de la primera lista de la estructura. Si son diferentes significa que la *sentence* que analizamos no puede ser generada por nuestra gramática.

Si son iguales, el análisis es posible y la palabra se contrasta con cada uno de los primeros átomos de las distintas listas que integran la estructura.

Cada uno de los siguientes términos de la oración a analizar es comparado sucesivamente con el contenido de los rectángulos en turno de las listas de la estructura. Así, puede llegarse a dos situaciones diferentes.

Una de ellas resulta cuando el rectángulo comparado contiene un átomo y éste es diferente del término de la oración. En este caso, la lista de la estructura deja de tenerse en cuenta.

La otra situación ocurre cuando el contenido del rectángulo comparado es diferente del término de la oración, por ser tal contenido una dirección (*address*). En este caso, el algoritmo se encarga de copiar enfrente del resto de la lista que está investigando la lista cuya dirección ha sido alcanzada en el proceso de análisis.

Así se continúa hasta, o bien alcanzar el final de una lista de la estructura, en cuyo caso la estructura oracional analizada es correcta, o bien alcanzar un átomo distinto al término de la oración en todas y cada uno de las listas que produce la gramática. En este caso la estructura oracional es incorrecta.

Ilustramos la explicación con un ejemplo. Procederemos al análisis de la siguiente oración:

The president the White House likes is ill.

La primera palabra de la oración (*the*) dejará las reglas definitorias de *sentence* del siguiente modo:

noun is adjective
adjective noun is adjective
noun noun-phrase verb is adjective
adjective noun noun-phrase verb is adjective.

La segunda palabra de la oración es *president.*

Dos de las listas serán descartadas y las otras restantes quedarán del siguiente modo:

is adjective
noun-phrase verb is adjective.

La siguiente palabra (*the*) hace descartar la lista que empieza por *is*. La otra lista comienza con un *class-name* y todas las listas que la definen se examinan comparando sus primeros átomos con *the*. Ello hace que en la memoria del ordenador se hayan dispuesto cuatro posibles estructuras para el resto de la oración:

noun verb is adjective
adjective noun verb is adjective
noun noun-phrase verb is adjective
adjective noun noun-phrase verb verb is adjective.

Continuando en esta linea podrá comprobarse que la oración que estamos analizando se corresponde, en cuanto a estructura sintáctica, con la segunda de las listas anteriores. El algoritmo que realiza el análisis lo exponemos en la figura 2.

Las *boxes* 1-4 sirven para leer la estructura (la gramática generativa) y almacenarla en memoria. Representación gráfica en la fig. 1.

La variable I (*box* 5) marca el comienzo del *working space*. Es decir, después de la variable 48 (*arrays* H y T) se construirán las nuevas listas.

La dirección (*address*) de la estructura total es T1 (*box* 5). Las estructuras (o listas) que se comparan con el término de la oración que se analiza van colocándose en L1 (*box* 20). En esta zona del algoritmo aparece el mecanísmo de construcción de nuevas listas (*box* 19) que simula el trabajo de la función CONS de los lenguajes de manipulación simbólica.

La variable C1 (*box* 7) se encarga de leer los sucesivos términos de la oración a analizar.

En el análisis, cada una de las listas de T1 se coloca en S1 (*box* 12). Si el contenido a investigar de S1 es un átomo, éste se compara con el término de la oración que se analiza. Si son desiguales la información contenida en ese momento del proceso en S1, deja de tenerse en cuenta. Si, por el contrario, son iguales, el análisis es posible. La "cola" de S1 se añade a L1 (*box* 19), a menos que sea cero; si así fuera, el análisis habría finalizado y la oración habría sido considerada correcta por el ordenador.

Si el contenido en turno de investigación en S1 no es un átomo (*box* 13), deberá ser la dirección (*address*) de una lista que representa un *class-name* (en nuestro caso, *noun-phrase*). Cada una de las reglas que la definen es antepuesta al resto de S1 y toda la estructura resultante antepuesta a su vez a T1 (*boxes* 24 y 25). Para realizar este trabajo se hace uso de distintas variables auxiliares: U1, S2, S3, etc. y del mecanísmo de una pila o *stack* K.

Debemos señalar que este proceso recursivo aparece en la zona derecha del diagrama y constituye uno de los mecanísmos suministrados normalmente por el compilador en el proceso de programas escritos en lenguajes de listas.

El trabajo de anteponer estructuras de listas a otras ya existentes se programa en los lenguajes especiales a base de la función recursiva APPEND.

Cuando el control regresa después de cada ciclo a la *box* 6, T1 contiene todas las estructuras sintácticas posibles y, además, dispuestas para el análisis del siguiente término de la oración.

El estudio detallado de las estructuras de lista que se crean durante el proceso de algoritmo puede realizarse con ayuda de un *trace* a base de *displays* situados inmediatamente a continuación de los mecanísmos de construcción de listas que aparece en el diagrama (*boxes* 19, 26 y 24).

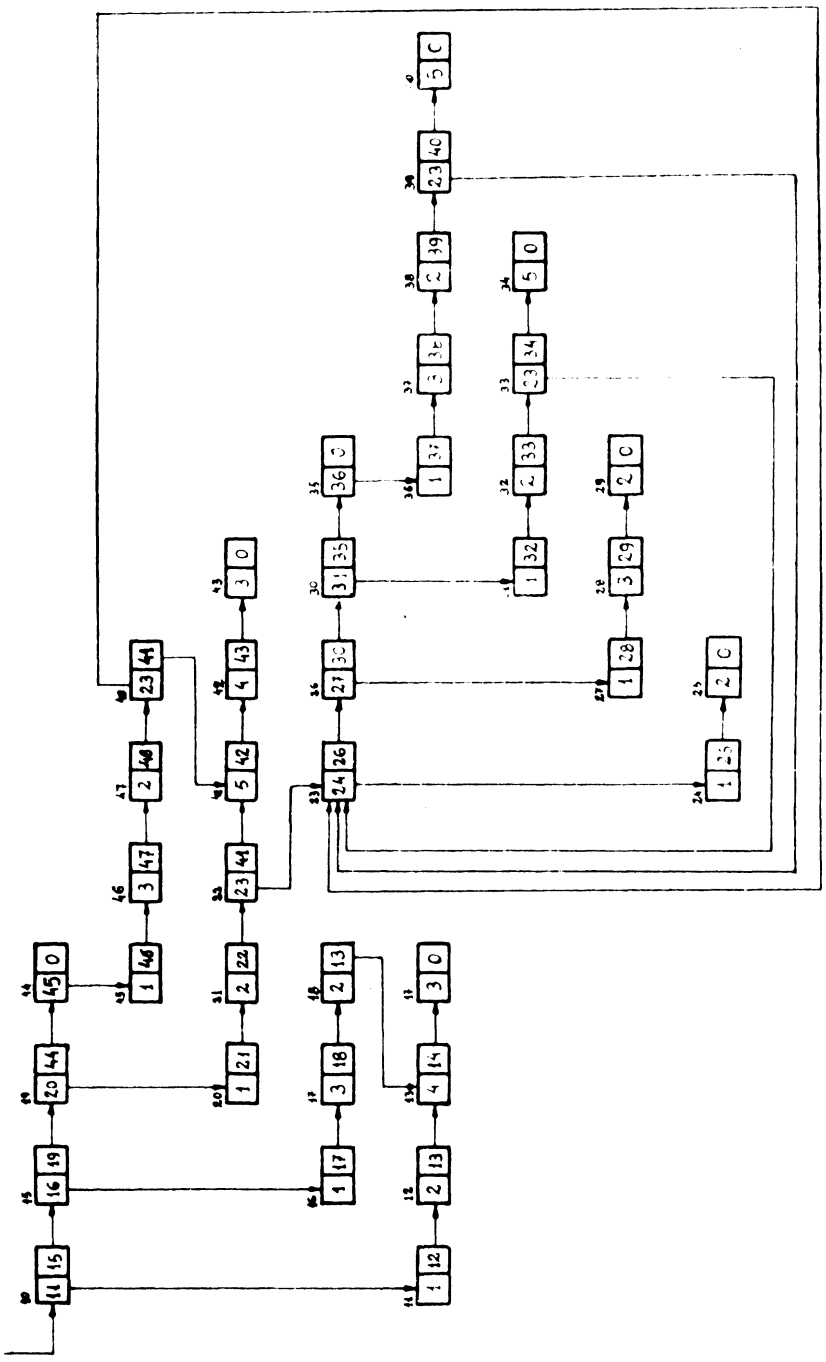

Figura 1

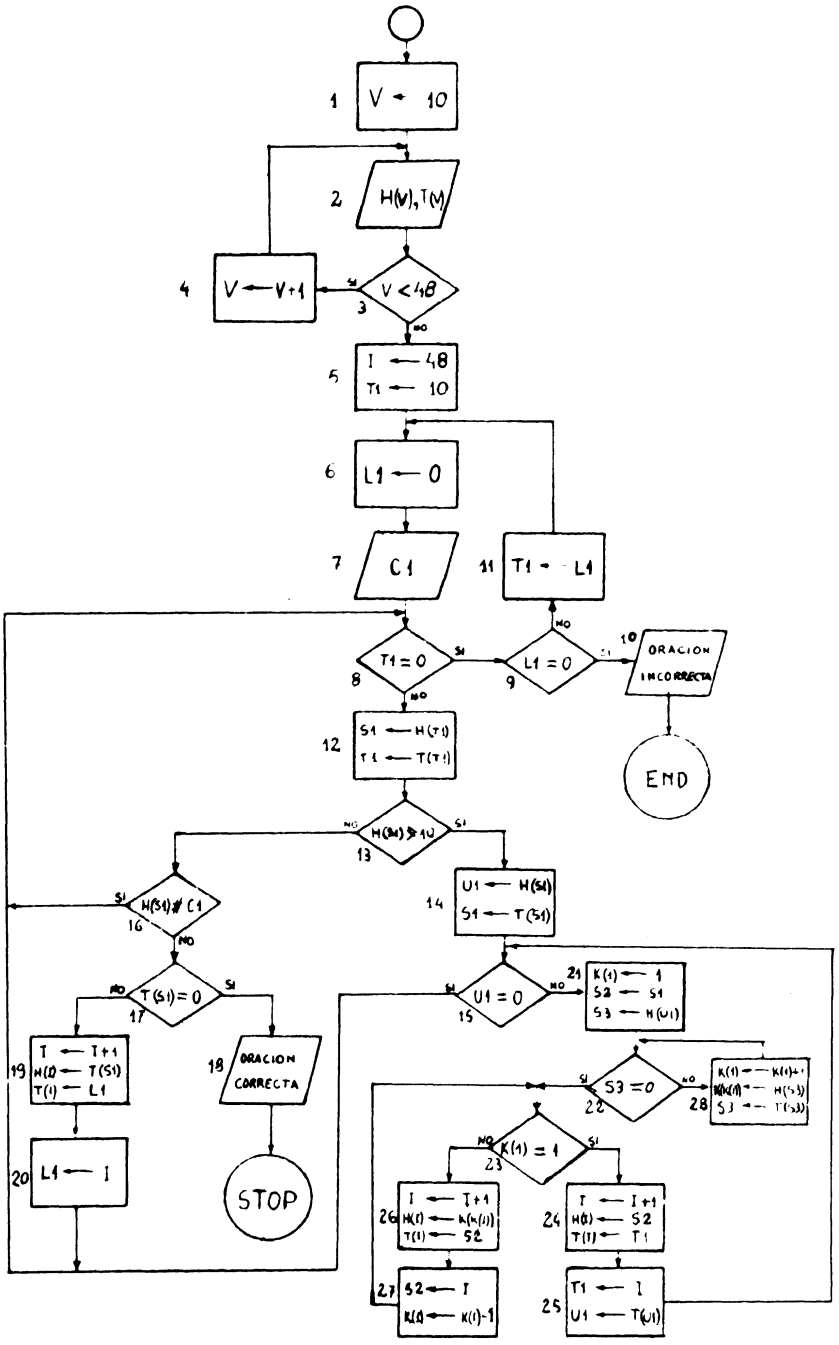

Figura 2

BIBLIOGRAFIA

BARRON, D.W.
Recursive Techniques in programming
Londres (1968), Macdonald.

CHOMSKY, N.
Estructuras Sintácticas
México (1974), Siglo XXI.

FOSTER, J.M.
List processing
Londres (1967), Macdonald.

KUNO, S. y OETTINGER, A.
Multiple path syntactic analyser
en "Information Processing" 62
Amsterdam (1963), North-Holland.

MCCARTHY, J.
Recursive functions of symbolic expressions and their computation by machine
(1960).

ROSEN, S.
Programming Systems and Languages
Nueva York (1967), McGraw-Hill.

MCCARTHY, J.
The LISP 1.5 Programmer's Manual
Cambridge (1962), (Massachusetts), MIT Press.