UM GERADOR DE ANALISADORES SINTÁTICOS LL(1)

Angela Maria Carvalho Souza

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.)

Aprovada por:

José Lucas Mourão Rangel Netto
(Presidente)

Sueli Mendes dos Santos

Luis Fernando Loureiro Legey

RIO DE JANEIRO, RJ - BRASIL

JULHO 1980

FICHA CATALOGRÁFICA

SOUZA, ANGELA MARIA CARVALHO

Um Gerador de Analisadores Sintáticos LL(1) |Rio de Janeiro| 1980.

V,120 p., 29,7 cm (COPPE-UFRJ, M.Sc. Engenharia de Sistemas e Computação, 1980)

Tese - Universidade Federal do Rio de Janeiro, Faculdade de Engenharia.

1.Análise Sintática I.COPPE/UFRJ II.Um Gerador de Analisadores Sintáticos LL(1).

A meus pais

RESUMO

Este trabalho descreve um gerador de tabelas diretoras do analisador sintático LL(1) para gramáticas livre de contexto. Se a gramática for LL(1) o resultado fornecido pelo gerador é a tabela diretora do analisador sintático da gramática dada ou da gramática transformada que se obtem por eliminação de recursão à esquerda e fatoração. Caso contrário o gerador indica todos os conflitos encontrados. Estão incluídos exemplos e o modo de utilização do programa escrito em Algol Extendido B6700 do NCE da UFRJ.

ABSTRACT

We describe here an LL(1) parser generator for context-free grammars. If the input grammar is LL(1), the output is the parsing table. If necessary, the generator will perform grammar modifications such as elimination of left recursion and factoring. If the transformed grammar is not LL(1) yet, the generator will produce a listing of all existing conflicts. Included are examples and description of use of the generator, written in Extended Algol of the NCE-UFRJ B6700 computer.

ÍNDICE

I - INTRODUÇÃO

Esta tese descreve um gerador de analisadores sintáticos que usa técnicas LL(1) para construção de analisadores "top-down". As técnicas LL(1) foram apresentadas por FOSTER[2] em 1968 e receberam um tratamento teórico de KNUTH[3] em 1971.

Normalmente, a primeira coisa a ser especificada no projeto de um compilador é o método de análise sintática a ser utilizado. Isso se deve ao fato de que via de regra, o analisador sintático dirige o compilador, solicitando "tokens" ao analisador léxico, e em linguagem pitoresca "servindo de cabide para pendurar a semântica",

O fato é que a análise sintática é uma das fases de compilação mais usadas e melhor compreendidas e para a qual existem teorias específicas bem construídas.

Por essa razão, raramente se utilizam hoje em dia, para o projeto de compiladores, técnicas manuais para a construção do analisador sintático, preferindo-se em geral transferir para um programa de computador a carga de descobrir como as diversas construções específicas da gramática de uma linguagem de programação podem ser reconhecidas.

Ao analisador sintático assim construído, (em geral constituído por um programa-padrão e tabelas específicas da gramática) acrescentamos as chamadas do analisador léxico e das rotinas semânticas, e o que mais se torne necessário para formar o compilador.

O texto desta tese foi dividido em capítulos cujo con

teúdo descrevemos a seguir.

Neste primeiro capítulo fazemos uma pequena apresentação e a seguir mostramos o que pode ser encontrado nos outros capítulos.

No capítulo II introduzimos definições básicas e notações que se fazem necessárias para a compreensão do desenvolvimento teórico do processo geral de análise sintática, e das linguagens e gramáticas LL apresentadas no capítulo III.

O sistema responsável pela construção do gerador e sua organização geral são descritos no capítulo IV. Para melhor compreensão o capítulo foi dividido em módulos, onde cada módulo é tratado em separado.

No capítulo V mostramos por meio de uma gramática escolhida como exemplo, como pode ser utilizado o sistema e o que ele pode oferecer ao usuário.

As conclusões e extensões possíveis são apresentadas no capítulo VI.

A listagem do sistema pode ser encontrada no apêndice.

II - DEFINIÇÕES E CONCEITOS FUNDAMENTAIS

Este capítulo procura introduzir definições básicas da teoria de gramáticas e linguagens de programação bem como algumas notações associadas com a teoria matemática de linguagens formais.

Alguns conceitos e notações, tais como alfabeto, sequência de símbolos, serão considerados como primitivos para nossos fins.

Um alfabeto é normalmente denotado por uma letra grega maiúscula; as mais comuns são Σ, N, Δ, Γ.

Definição 1: O conjunto de todas as sequências construídas com símbolos de um alfabeto qualquer Σ será denotado por Σ^*.

A sequência vazia ε é a sequência de zero símbolos.

O conjunto de todas as sequências não vazias construídas com símbolos de um alfabeto qualquer Σ será denotado por Σ^+.

Definição 2: O comprimento $|x|$ de uma sequência x é o número de símbolos que compõem x.

Por exemplo, seja $\Sigma = \{A,\ldots,Z\}$ então $|BEGIN| = 5$.

Se por outro lado considerarmos Σ como sendo o alfabeto do ALGOL {begin, end, case,..., A,...,Z, 0, ...,9, (,...}, então |begin| = 1.

Definição 3: Uma linguagem sobre o alfabeto Σ é um subconjunto de Σ^*, isto é, um conjunto de sequências de símbolos de Σ.

As linguagens podem ser vazias, não vazias, finitas, infinitas.

Seja $\Sigma = \{0, 1\}$. Então

ϕ

Σ^*

Σ^+

$\{\varepsilon\}$

$\{0, 1\}$

$\{x \in \Sigma^+ \mid$ o último símbolo de x é 1$\}$ são linguagens sobre $\{0, 1\}$.

A definição 3 introduz um conceito de linguagem que pode modelar linguagens naturais ou linguagens artificiais. Entre estas últimas se situam as lingugens de programação. A escolha do alfabeto apropriado é fundamental.

Um dos dispositivos usados para especificar uma linguagem finita é a gramática. Vejamos agora a definição de gramática.

Definição 4: Uma gramática é uma quádrupla $G = (N, \Sigma, P, S)$, onde:

- N é um alfabeto, cujos símbolos são os não-terminais de G.
- Σ é um alfabeto, cujos símbolos são os terminais de G.

- $N \cap \Sigma = \phi$ ou seja, N e Σ são alfabetos distintos.
- S é um não-terminal de G($S \in N$), o símbolo inicial de G.
- P é um conjunto finito de produções (regras) da forma $\alpha \rightarrow \beta$ onde α e β são sequências de símbolos terminais e não-terminais, sendo obrigatória a presença de pelo menos um não-terminal em α.

Convenção: Usaremos as seguintes notações para representar símbolos e sequências:

- letras maiúsculas do começo do alfabeto, tais como, A, B,... representam símbolos não-terminais (N).

- letras minúsculas do começo do alfabeto, tais como, a, b,... representam símbolos terminais (Σ).

- letras maiúsculas próximas do fim do alfabeto, tais como, X, Y,... representam símbolos da gramática, isto é, terminais e/ou não-terminais ($\Sigma \cup N$).

- letras minúsculas próximas do fim do alfabeto, tais como, u, v,...,z, representam sequências de terminais (Σ^*).

- letras gregas minúsculas, tais como, α, β, γ,... representam sequências de símbolos da gramática, isto é, sequências de terminais e/ou não-terminais $(\Sigma \cup N)^*$. Assim uma produção genérica pode aparecer como $A \rightarrow \alpha$, indicando que há um não-terminal A à esquerda da seta (o lado direito da produção) e

uma sequência de símbolos da gramática α à direita da seta (o lado direito da produção).

- Se $A \to \alpha_1, A \to \alpha_2, \ldots, A \to \alpha_k$ são todas produções com A como lado esquerdo, podemos escrever $A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_k$ e chamar $\alpha_1, \alpha_2, \ldots, \alpha_k$ de <u>alternativas</u> de A.

- o lado esquerdo da primeira produção é sempre o símbolo inicial da gramática.

Definição 5: Seja $G = (N, \Sigma, P, S)$. Dizemos que uma sequência $\gamma\ \alpha\ \delta$ <u>deriva diretamente em G</u> a sequência $\gamma\ \beta\ \delta$ se P contém a regra $\alpha \to \beta$.

Em símbolos: $\gamma\ \alpha\ \delta \Rightarrow_G \gamma\ \beta\ \delta$

Dizemos que a sequênica α <u>deriva em G, em zero ou mais passos</u> a sequência β se existem um inteiro $n \geq 0$ e sequências $\alpha_0, \alpha_1, \ldots, \alpha_n$ tais que:

$\alpha_0 = \alpha$

$\alpha_n = \beta$

$\alpha_{i-1} \Rightarrow_G \alpha_i$ para $i = 1, \ldots, n$

Em símbolos: $\alpha \Rightarrow_G^* \beta$

Dizemos que uma sequênica α <u>deriva em G, em um ou mais passos</u> a sequência β se $\alpha \Rightarrow_G^* \beta$, sendo $n \geq 1$ na definição acima.

Em símbolos: $\alpha \Rightarrow_G^+ \beta$

Caso não seja necessário especificar a gramática G, os símbolos $\Rightarrow$, $\Rightarrow^*$, $\Rightarrow^+$ serão usados.

Podemos agora introduzir formalmente o conceito de linguagem definida por uma gramática.

Definição 6: Seja $G = (N,\Sigma,P,S)$. A linguagem $L(G)$ é definida como:

$$L(G) = \{x \in \Sigma^* \mid S \Rightarrow_G^* x\}$$

Definição 7: Uma <u>forma sentencial</u> de $G = (N,\Sigma,P,S)$ é uma sequência α tal que $S \Rightarrow_G^* \alpha$.

Vamos agora introduzir a hierarquia de gramáticas e de linguagens, devida originalmente a Chomsky.

Definição 8: Uma gramática G é:

<u>tipo 0</u> se simplesmente satisfaz a definição 4.

<u>tipo 1</u> (<u>sensível ao contexto</u>) se todas as suas produções são da forma $\alpha \to \beta$ onde $|\alpha| \leq |\beta|$.

<u>tipo 2</u> (<u>livre de contexto</u>) se todas as suas produções são da forma $A \to \beta$, onde $|\beta| \geq 1$ e $A \in N$.

<u>tipo 3</u> (<u>regular</u>) se todas as suas produções são da forma $A \to aB$ ou $A \to a$ onde A e $B \in N$ e $a \in \Sigma$.

Definição 9: Uma linguagem L é do tipo $i = 0, 1, 2, 3$, se existe uma gramática G do mesmo tipo i tal que $L = L(G)$.

Podemos observar que, de acordo com a definição 9, a

sequência vazia não pode pertencer a nenhuma linguagem tipo 1, 2 ou 3. Para incluir a sequênica vazia, as definições de gramáticas tipo 1, 2 e 3 são modificadas na forma abaixo:

- permitiremos em uma gramática tipo 1 (sensível ao contexto) uma produção da forma $S \rightarrow \varepsilon$, onde S é o símbolo inicial, com a condição de que S não apareça em nenhum lado direito de produção.

- liberaremos o uso de regras da forma $A \rightarrow \varepsilon$ em gramáticas livre de contexto e regulares (tipo 2 e 3).

Das quatro classes de gramáticas na hierarquia de Chomsky, as gramáticas livre de contexto são as mais importantes em termos de aplicações de linguagens de programação.

Uma gramática livre de contexto pode ser usada para especificar a maioria das estruturas sintáticas de uma linguagem de programação. Daqui para frente, linguagem indicará linguagem livre de contexto.

III - LINGUAGENS E GRAMÁTICAS LL

Antes de entrarmos em detalhe no estudo do gerador de analisadores sintáticos que é o objetivo principal desta tese, vamos apresentar aqui as noções fundamentais para a compreensão do processo de análise sintática.

Por análise sintática, compreendemos o problema de, (dadas uma gramática livre de contexto G e uma sequência x de sua linguagem), determinar como a sequência x pode ser derivada a partir do símbolo inicial de G. Frequentemente considera-se também em análise sintática, o problema de mostrar que uma dada sequência x não pertence à linguagem da gramática G.

Uma distinção fundamental existe entre o método de análise sintática LL(1), que é um método top-down, e outros métodos frequentemente encontrados, que são métodos bottom-up.

No método top-down, a análise é feita procurando em cada passo determinar que produção deve ser usada a seguir. Se a sentença x é examinada da esquerda para a direita, normalmente o processo segue uma derivação à esquerda.

Exemplo 1. Sejam a gramática G.

E → E + T	1
E → T	2
T → T * F	3
T → F	4
F → (E)	5
F → a	6

e a sentença x = (a + a) * a. A árvore de derivação é mostrada na figura III.1 e a simulação do processo de análise é mostrado no quadro da figura III.2, onde p é a sequência de números de produções usadas na derivação à esquerda. Os números encontrados na árvore indicam a ordem em que as expansões são realizadas durante a análise.

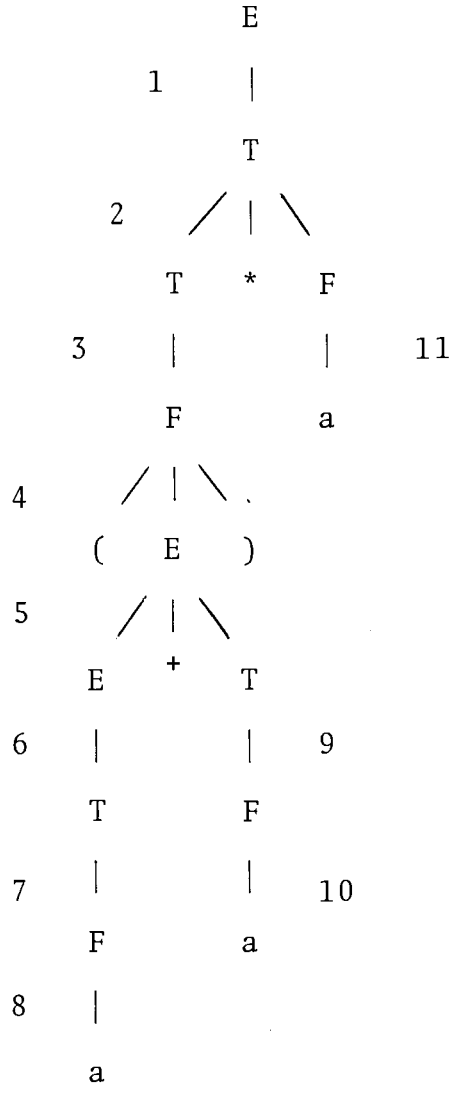

Fig. III.1

TEXTO GERADO		P	TEXTO A ANALISAR	
	E			(a + a) * a
	T	2		(a + a) * a
	T * F	3		(a + a) * a
	F * F	4		(a + a) * a
	(E) * F	5		(a + a) * a
(	E) * F	-	(	a + a) * a
(	E + T) * F	1	(	a + a) * a
(	T + T) * F	2	(	a + a) * a
(	F + T) * F	4	(	a + a) * a
(	a + T) * F	6	(	a + a) * a
(a	+ T) * F	-	(a	+ a) * a
(a +	T) * F	-	(a +	a) * a
(a +	F) * F	4	(a +	a) * a
(a +	a) * F	6	(a +	a) * a
(a + a	) * F	-	(a + a	) * a
(a + a)	* F	-	(a + a)	* a
(a + a) *	F	-	(a + a) *	a
(a + a) *	a	6	(a + a) *	a
(a + a) * a		-	(a + a) * a	

Fig. III. 2

No método bottom-up procuramos em cada passo determinar que redução deve ser efetuada, partindo da sentença x, e procurando obter o símbolo inicial de G. Neste caso, mantendo-se o sentido de exame da sentença x da esquerda para a direita, obteremos uma derivação à direita construída em ordem inversa.

A figura III.3 mostra na árvore de derivação da sentença x, do exemplo 1, usando o método bottom-up, a ordem em que as reduções são efetuadas.

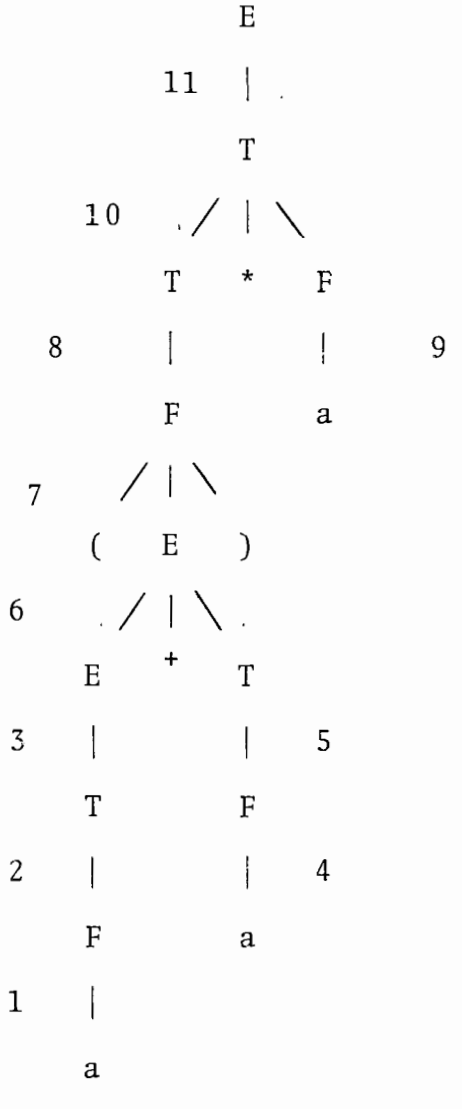

Fig. III.3

Vejamos, agora, com detalhes, o que acontece com a análise top-down.

Exemplo 2: Consideremos a gramática abaixo que descreve um bloco tipo ALGOL.

B → b D ; C e

D → d | d ; D

C → c | c ; C

(onde b, C, D, e correspondem a begin, "comando", "declaração", end).

Consideremos também a sentença: b d ; d ; c ; c e como entrada do analisador.

A análise é feita comparando as possibilidades, ou formas sentenciais possíveis, com a sentença ou texto fonte. A primeira possibilidade é o símbolo inicial da gramática, e as novas possibilidades são produzidas por substituição dos símbolos não-terminais mais à esquerda por suas possíveis expansões. Aquelas possibilidades que não correspondem ao texto são rejeitadas, deixando as outras para comparações futuras, e assim por diante. Os estados sucessivos da análise são dados no quadro da figura III.4.

Resumindo, as regras da análise vemos que:

- o estado inicial tem o símbolo inicial como forma sentencial única e o texto fonte.
- qualquer forma sentencial que comece com um símbolo não-terminal é substituída por tantas quantas forem necessárias, cada uma sendo a forma sentencial original com o símbolo não

-terminal substituído por uma de suas expansões.

- se todas as formas sentenciais posssíveis começam com um símbolo terminal, cada um desses símbolos é comparado com o símbolo mais à esquerda do texto. Se o símbolo não pertence ao texto, a forma sentencial correspondente é rejeitada, senão o símbolo é retirado da mesma e passa-se para o próximo estado, onde o texto também perde seu primeiro caráter.
- a análise termina quando uma possibilidade e o texto tornam-se vazios simultâneamente. Se várias possibilidades tornam-se vazias neste ponto a linguagem é ambígua, e se nenhuma possibilidade torna-se vazia, o texto não é da linguagem. Este também é o caso se nenhuma possibilidade existe em algum momento.
- recursão à esquerda acarreta "loop" ao analisador, uma vez que torna-se impossível deixar de obter formas sentenciais começando com um símbolo não-terminal.

	POSSIBILIDADES	TEXTO
1	B	b d ; d ; c ; c e
2	b D ; C e	b d ; d ; c ; c e
3	D ; C e	d ; d ; c ; c e
4	d ; C e d ; D ; C e	d ; d ; c ; c e
5	; C e ; D ; C e	; d ; c ; c e
6	C e D ; C e	d ; c ; c e
7	c e c ; C e d ; C e d ; D ; C e	d ; c ; c e
8	; C e ; D ; C e	; c ; c e
9	C e D ; C e	c ; c e
10	c e c ; C e d ; C e d ; D ; C e	c ; c e
11	e ; C e	; c e
12	C e	c e
13	c e c ; C e	c e
14	e ; C e	e
15		

Fig. III.4

Consideremos o estado 4 da figura III.4. A ação determinante para o estado 5 foi o primeiro símbolo de cada uma das duas possibilidades e o símbolo do texto. Esta correspondência, no entanto, não é um processo instantâneo, uma vez que uma declaração consiste de vários caracteres. As duas possibilidades permanecem durante a análise da declaração completa, que é por conseguinte analisada duas vezes. Cada comando é também analisado duas vezes e este número é dobrado cada vez que os comandos são referenciados. Este desperdício pode muitas vezes ser eliminado escolhendo-se uma gramática diferente, para a qual só existe uma possibilidade por vez.

Exemplo 3: Seja a gramática abaixo:

B → b D ; C e

D → d G

G → ε | ; D

C → c H

H → ε | ; C

Se o analisador for mantido, ele trabalhará com essa gramática da seguinte maneira:

Possibilidades	Texto
B	b d ; d ; c ; c e
b D ; C e	b d ; d ; c ; c e
D ; C e	d ; d ; c ; c e
d G ; C e	d ; d ; c ; c e
G ; C e	; d ; c ; c e

e assim por diante. A declaração é analisada somente uma vez.

É possível obter uma gramática para a qual o analisador pode sempre reduzir o número de formas sentenciais possíveis para uma.

Informalmente, se o analisador puder tomar uma decisão inspecionando somente um caracter do texto de entrada, o caracter mais à esquerda, dizemos que, a gramática e o analisador são LL(1). Se k caracteres são necessários para obter a decisão a gramática e o analisador serão LL(k).

O analisador para este tipo de gramática é top-down e determinístico.

Estes analisadores são chamados LL porque pesquisam a entrada da esquerda para à direita (Left-to-right) e constroem uma derivação à esquerda (Leftmost).

Neste ponto passaremos a apresentar as condições para que uma gramática seja LL(1), antes porém definiremos alguns conceitos.

Consideremos as seguintes porduções para um não-terminal A.

$$A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$$

Definição 1: O conjunto de símbolos que podem ocorrer à esquerda de qualquer sequência gerada por uma sequência não vazia α de terminais e não-terminais é chamada de "símbolos iniciadores" de α e é definido como:

$$I(\alpha) = \{a \in \Sigma \mid \alpha \Rightarrow^* a\beta \in (\Sigma \cup N)^*\}$$

Definição 2: O conjunto de símbolos que podem ocorrer imediatamente à direita de um não-terminal A em alguma forma sentencial é chamado de "símbolos seguidores" de A. Se A é o símbolo mais à direita em uma forma sentencial então $ (um símbolo novo) é incluido no conjunto de símbolos seguidores de A. ($ representa na prática um indicador de fim de arquivo)

$$S(A) = \{a \in \Sigma \cup \{\$\} \mid X\$ \Rightarrow^* \beta A a \delta,\ \beta \in (\Sigma \cup N)^*,\ \delta \in (\Sigma \cup N \cup \{\$\})^*,\ A \in N\}$$

Definição 3: O conjunto de símbolos diretores para uma expansão α de um não-terminal A é definido como:

$$SD(A, \alpha) = \{a \mid \alpha \neq \varepsilon \text{ e } a \in I(\alpha) \text{ ou } (\alpha \Rightarrow^* \varepsilon \text{ e } a \in S(A))\}$$

A condição necessária e suficiente para uma gramática ser LL(1) é que os símbolos diretores correspondentes a diferentes expansões de cada não terminal estejam em conjuntos disjuntos.

A justificativa desta condição é a seguinte:

- a condição é necessária porque se um símbolo aparece em dois conjuntos de símbolos diretores o analisador não pode decidir que expansão aplicar sem informação futura.
- a condição é suficiente, porque o analisador pode sempre escolher uma expansão em função do símbolo dado e esta é sempre a certa. Se o símbolo não está contido em nenhum dos con

juntos de símbolos diretores, o texto não é da linguagem e há um erro.

Analisador LL(1)

Uma gramática LL(1) permite o uso de formas especiais de analisadores. Uma desses é o método de descida recursiva, no qual cada não-terminal corresponde a um procedimento do analisador sintático.

Exemplo 4: Considere a gramática LL(1)

B → b d ; X c Y e

X → d ; X | ε

Y → ; c Y | ε

Um analisador para esta gramática seria:

```
procedure B ;

begin check ('b') ;
      check ('d') ;
      check (';') ;
      X ;
      check ('c') ;
      Y ;
      check ('e') ;

end ;

procedure X ;

begin if    caracter = 'd'
      then  begin check  ('d') ;
                  check  (';') ;
                  X
            end ;

end ;

procedure Y ;

begin if    caracter = ';'
      then  begin  check (';') ;
                   check ('c') ;
                   Y
            end ;

end ;
```

onde check é uma macro que confirma se o caracter corrente (o caracter mais à esquerda do texto fonte) é o mesmo que o parâmetro atual e desloca um caracter do texto fonte. Se o caracter corrente não é confirmado, o analisador acusa um erro.

Uma outra estrutura possível para o analisador LL(1) é um programa padrão, que tem acesso a uma tabela, onde ficam reunidas as informações sobre a gramática. Neste caso, uma alteração na gramática corresponde apenas a uma alteração nos dados constantes da tabela. Normalmente este tipo de analisador é chamado dirigido pela tabela (table-driven parser). O analisador preditivo descrito a seguir é deste tipo, dessa forma, a necessidade de uma linguagem que permita chamadas recursivas de procedimentos é afastada.

O analisador preditivo mantém uma pilha, (e é por este motivo que se dispensa a recursividade) e sua estrutura básica está na figura III.5.

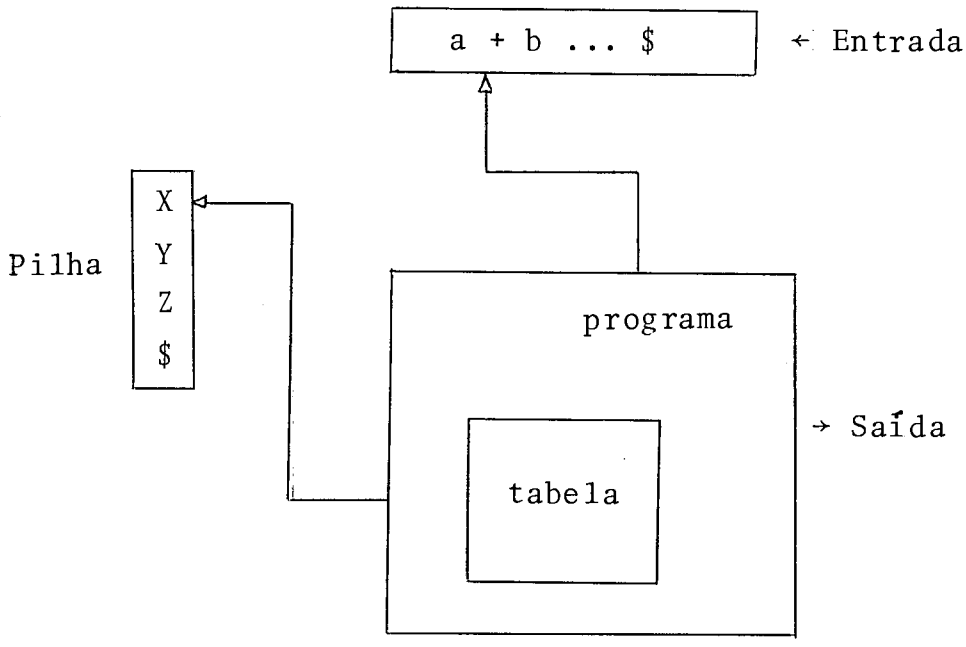

Fig. III.5

A entrada contém a cadeia para ser analisada, seguida de $, a marca de fim. A pilha contém uma sequência de símbolos da gramática, precedida de $, a marca de final da pilha. Inicialmente, a pilha contém o símbolo inicial da gramática precedido de $. A tabela de análise, definida como: $T|A,a| = j$ se e somente se $j : A \rightarrow \alpha$ e $SD(A,\alpha) = a$, é uma matriz, onde A é um não-terminal e a é um terminal ou o símbolo $.

O analisador é controlado pelas ações que seguem. O programa determina X, o símbolo no topo da pilha, e a, o símbolo corrente da entrada. Estes dois símbolos determinam a ação do analisador. Existem três possibilidades:

1. Se $X = a = \$$, o analisador para indicando a conclusão da análise com sucesso.

2. Se $X = a \neq \$$, o analisador retira X da pilha e avança o ponteiro da entrada para o próximo símbolo da entrada.

3. Se X é um não-terminal A, o programa consulta a entrada $T|A,a|$ da tabela. Esta entrada será uma produção do não-terminal A da gramática, (na prática, o número da produção) ou uma entrada de erro. Se $T|A,a| = \{A \rightarrow \alpha\}$, o analisador substitui A no topo da pilha por α. Se $T|A,a|$ = erro, isto é, entrada em branco, o analisador chama uma rotina de recuperação de erro.

Exemplo 5: Considere a gramática LL(1) do exemplo 4. A tabela de análise para esta gramática encontra-se na figura III.6. Brancos são entradas de erro. (A ma

neira como estas entradas são selecionadas será mostrada no próximo capítulo). Com a entrada do exemplo 2 o analisador fará a sequência de movimentos mostrada na figura III.7.

	b	d	;	c	e	$
B	B→bd;XcYe					
X		X→d;X		X→ε		
Y			Y→;cY		Y→ε	

Fig. III.6

Para algumas gramáticas a tabela T possui entradas que são definidas multiplamente. Poe exemplo, se a gramática é recursiva à esquerda ou ambígua, então a tabela terá pelo menos uma entrada com definição múltipla.

Se a gramática não for LL, $SD(A,\alpha) \cap SD(A,\alpha') \neq \phi$ e portanto $T|A,a|$ não pode ser definido inequivocamente para todos os a's.

PILHA	ENTRADA	SAÍDA
$ B	b d ; d ; c ; c e $	
$ e Y c X ; d b	b d ; d ; c ; c e $	B → b d ; X c Y e
$ e Y c X ; d	d ; d ; c ; c e $	
$ e Y c X ;	; d ; c ; c e $	
$ e Y c X	d ; c ; c e $	
$ e Y c X ; d	d ; c ; c e $	X → d ; X
$ e Y c X ;	; c ; c e $	
$ e Y c X	c ; c e $	
$ e Y c	c ; c e $	X → ε
$ e Y	; c e $	
$ e Y c ;	; c e $	Y → ; c Y
$ e Y c	c e $	
$ e Y	e $	
$ e	e $	Y → ε
$	$	

Fig. III.7

Exemplo 6: Seja a gramática definida por:

S → i C t S S' | a

S' → e S | ε

C → b

A tabela de símbolos diretores para esta gramática é mostrada na figura III.8. Nela a entrada para |S',e| contém tanto S' → e S como S → ε.

	a	b	e	i	t	$
S	S → a			S → iCtSS'		
S'			S' → ε S' → eS			S' → ε
C		C → b				

Fig. III.8

A gramática do exemplo 6 é ambígua e a ambiguidade é manifestada na escolha de que produção usar quando um e (else) aparece. Esta ambiguidade pode ser resolvida se escolhermos S' → eS, o que corresponde a associarmos o else ao último if.

Podemos assim informalmente dizer que uma gramática LL(1) possui uma tabela de análise que não possui entradas múltiplas.

IV - DESCRIÇÃO DO SISTEMA

Neste capítulo descreveremos o gerador de analisadores sintáticos LL(1), cuja implementação foi o objetivo de nossa tese, os algoritmos e os métodos utilizados para alcançar este fim.

O sistema é voltado para o uso do computador Burroughs B6700 e a linguagem de programação usada é o ALGOL.

Fizemos uso de estrutura de dados com manipulação parcial de palavras, procurando sempre que possível usar os 48 bits acessíveis.

O sistema, a princípio, funciona em batch, sendo simples a sua adaptação para on-line.

São fornecidos ao usuário três módulos distintos:

- um codificador cuja função é fornecer à gramática uma representação interna,
- um testador da condição LL(1),
- um módulo que tenta transformar gramáticas não LL(1) em gramáticas equivalentes LL(1),

além de algumas rotinas auxiliares de impressão de resultados intermediários de cada um dos módulos acima, e o gerador propriamente dito.

A organização geral do sistema é mostrada no fluxograma da figura IV.1.

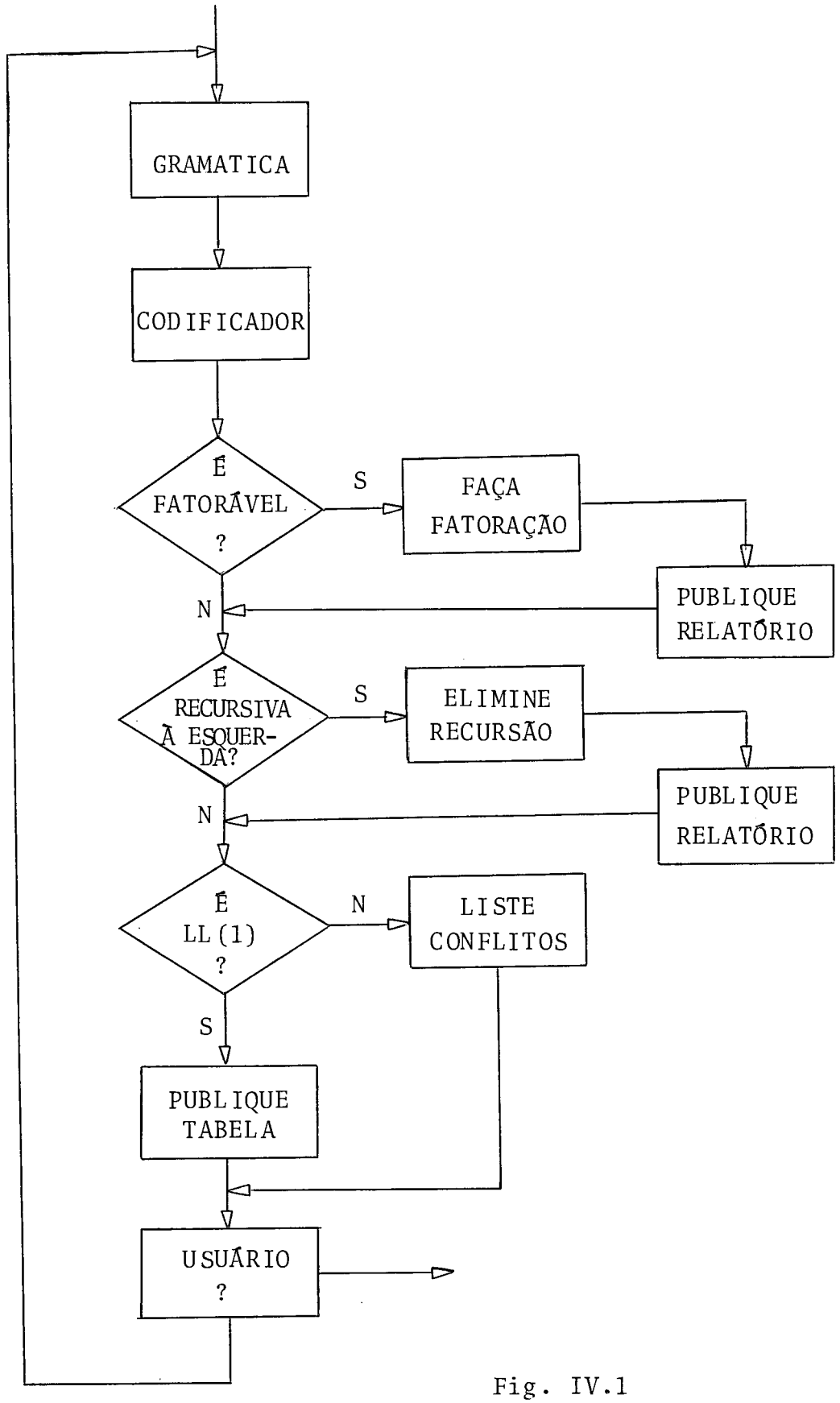

Fig. IV.1

Obs.: Neste fluxograma não mostramos as saídas anormais do sistema.

IV.1 - CODIFICADOR

Neste módulo, a gramática cuja representação se encontra no formato de entrada, sofre uma análise léxica e sintática, recebe uma representação interna e é gerada a tabela de símbolos para os não-terminais e terminais.

A área de memória estimada para a gramática padrão apresenta o seguinte "default":

200 não-terminais
500 terminais
comprimento médio do lado direito da produção = 5
produção média por não-terminal = 4

Exemplo: Seja a gramática:

```
1   EXPRESSÃO = EXPRESSÃO '+' TERMO
2             | TERMO ;
3   TERMO  =  TERMO  '*'  FATOR
4          |  FATOR
5          ;
6   FATOR  =  '('  EXPRESSÃO  ')'
7          |  'A'  ;
```

Sua representação interna é mostrada na figura IV.2.

Para cada não-terminal da gramática é associado uma palavra na estrutura sequencial PONT, que aponta para a primeira produção desse não-terminal.

A estrutura ligada NO é formada por três campos: o primeiro ocupa os bits 47 à 32, o segundo os bits 31 à 16 e o terceiro os bits 15 à 0.

Quando um NO é usado para representar o lado esquerdo de uma produção o primeiro campo aponta para a produção seguinte do mesmo não-terminal, o segundo campo contém o número do cartão onde essa produção é iniciada, e o terceiro campo aponta para a representação do lado direito da produção.

Na representação do lado direito da produção o primeiro campo não é utilizado, o segundo campo contém o código do símbolo terminal ou não-terminal, e o terceiro campo aponta para a representação do próximo símbolo.

A escolha de uma estrutura ligada é decorrente da necessidade de modificações nas produções dos não-terminais já existentes, tais como retirada ou inclusão de símbolos ou mesmo de produções, bem como criação de novos não-terminais.

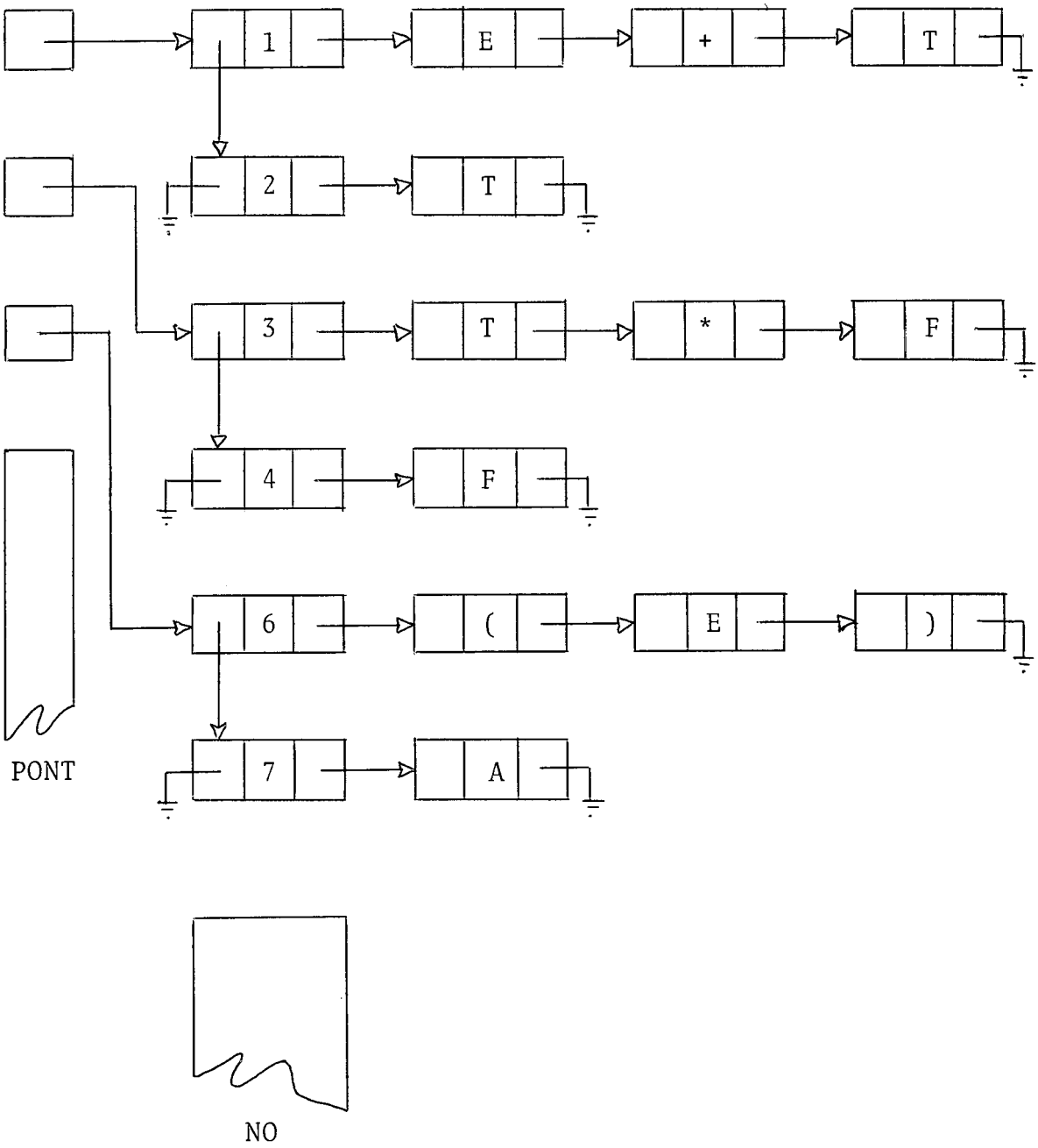

Fig. IV.2

FORMATO DE ENTRADA

Representamos uma gramática livre de contexto na seguinte forma:

```
GLC = ESPAÇO                                          # glc vazia
    | GLC  REGRA ESPAÇO ;
REGRA  = PARTE*ESQUERDA '=' ALTERNATIVAS ';'
       ;
ALTERNATIVAS  = PARTE*DIREITA
              | ALTERNATIVAS '|' PARTE*DIREITA ;
PARTE*ESQUERDA  = NÃO*TERMINAL
                ;
PARTE*DIREITA   = ESPAÇO                  # lado direito sequencia vazia
                | PARTE*DIREITA TERMINAL ESPACO
                | PARTE*DIREITA NÃO*TERMINAL ESPACO ;
NÃO*TERMINAL  = IDENTIFICADOR
              ;
TERMINAL  =  CADEIA
          ;
IDENTIFICADOR  =  LETRA
               |  IDENTIFICADOR LETRA
               |  IDENTIFICADOR DIGITO
               |  IDENTIFICADOR ESTRELINHA ;
CADEIA  =  PLICA  QUALQUER*COISA PLICA
        ;
ESPACO  =  SEPARADOR
        |  ESPACO SEPARADOR ;
```

```
SEPARADOR = BRANCO
          | MUDALINHA
          | COMENTARIO ;
COMENTARIO  = JOGO*DA*VELHA BRANCO QUALQUER*COISA MUDALINHA
          ;
QUALQUER*COISA =                                  # uma parte direita vazia
               | QUALQUER*COISA CARACTER
               | QUALQUER*COISA JOGO*DA*VELHA 'N'          # mudalinha
               | QUALQUER*COISA JOGO*DA*VELHA 'Q'          # uma plica
               | QUALQUER*COISA JOGO*DA*VELHA  JOGO*DA*VELHA # jogo da velha
               ;
CARACTER  =  LETRA
          |  DIGITO
          |  OPERADOR
          |  BRANCO ;
LETRA  =  'A' | 'B' | ... | 'Z'
       ;
DIGITO =  '0' | '1' | ... | '9' ;
OPERADOR = '=' | '|' | ';' | '+' | '-' | '*' | '/' | '<'| '>' |
           '¬' | ',' | '.' | ':' | '(' | ')'
         ;
PLICA = '#Q'                              # convenção para o uso da plica
      ;
JOGO DA VELHA = '##'                 # convenção para o uso do jodo da velha
              ;
MUDALINHA  =  '#N'                    # convenção para o uso do mudalinha
           ;
BRANCO  =  ' '
        ;
ESTRELINHA = '*'
           ;
```

IV.2 - TESTADOR DA CONDIÇÃO LL(1)

Este módulo tem por objetivo testar a condição LL(1) para uma dada gramática. Ele é formado pelos geradores de:

- símbolos iniciadores
- símbolos seguidores
- símbolos diretores

O algoritmo que decide se uma gramática G é ou não LL(1) determina o conjunto de símbolos diretores para cada expansão de um não-terminal e em seguida aplica a condição LL(1).

Para determinar os conjuntos de símbolos diretores, primeiro encontramos os conjuntos de símbolos iniciadores e seguidores para cada não-terminal, como também a informação de que o não-terminal gera ou não a sequência vazia.

Algoritmo que determina se um não-terminal pode ou não gerar a sequência vazia - rotina SEQUÊNCIAVAZIA, página A8 do apêndice.

Este algoritmo necessita de uma cópia da gramática e um vetor V, com uma entrada por não-terminal na gramática. Os elementos de V possuem um dos três valores: sim, não ou indefinido, dizendo se o não-terminal pode ou não gerar a sequência vazia.

O algoritmo funciona da seguinte maneira:

1 - cada elemento de V é inicializado com "indefinido"

2 - durante um passo na gramática são executadas as seguintes ações:

a) - se qualquer expansão de um não-terminal é a sequência vazia, o correspondente elemento de V recebe o valor "sim" e todas as produções do não-terminal são eliminadas da gramática.

b) - qualquer produção contendo um símbolo terminal é eliminada da gramática. Se essa ação elimina todas as produções de um não-terminal, o valor correspondente de V recebe o valor "não".

3 - a gramática está agora limitada pela regra na qual os lados direito contém somente símbolos não-terminais. Com sucessivos passos na gramática obtemos as seguintes ações, onde cada lado direito é examinado:

a) - se a entrada correspondente de V tem o valor "sim", o símbolo é eliminado. Se isso leva à sequência vazia como lado direito, o não-terminal para o qual está é uma expansão, pode gerar a sequência vazia. A correspondente entrada de V torna-se "sim" e as produções do não-terminal são eliminadas.

b) - se a correspondente entrada de V tem o valor "não", a produção é eliminada. Se todas as produções do não-terminal são eliminadas dessa maneira, sua entrada em V recebe o valor "não".

4 - se durante um passo completo na gramática, nenhuma entrada de V é trocada e há ainda entradas indefinidas a gramática não é LL(1) e isto provoca uma terminação anormal do algo

ritmo.

A terminação anormal do algoritmo acima, é uma razão suficiente para dizer que a gramática não é LL(1), porque nesse caso, existe recursão à esquerda e não-terminais inúteis, uma vez que existe um certo número de produções consistindo somente de não-terminais não-vazios os quais não podem gerar sequências que não contenham não-terminais.

Para qualquer gramática que não contém não-terminais inúteis somos capazes de produzir o vetor V, indicando se um não-terminal pode ou não gerar a sequência vazia.

O próximo passo agora é o cálculo dos conjuntos de símbolos iniciadores.

IV.2.1 - CONJUNTOS DE SÍMBOLOS INICIADORES

O conjunto iniciador de cada não-terminal é calculado em uma matriz de bits composta de dois campos, mostrados na figura IV.3.

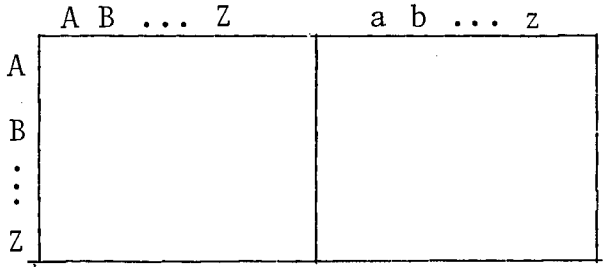

Fig. IV.3

Durante um passo da gramática os símbolos iniciadores imediatos são indicados na matriz.

Vejamos como isso acontece:

Por exemplo, na regra:

A → B c D | e

(B, A) e (e, A) recebem o valor 1

Se B pode gerar a sequência vazia (informação encontrada em V), c é também um iniciador de A, e (c,A) recebe o valor 1, e assim por diante.

A matriz de símbolos iniciadores imediatos não é suficiente como pode ser observado no exemplo seguinte.

Consideremos as produções:

A → B c D

B → b X

b é um iniciador de B e por sua vez um iniciador de A.

O fechamente transitivo, GRIES[10], da matriz de iniciadores imediatos fornece a matriz completa de símbolos iniciadores necessária para calcular os símbolos diretores.

A matriz de símbolos iniciadores, resultante do fechamento transitivo, permite um teste imediato de recursão à esquerda, visto que um 1 em sua diagonal principal indica que um não-terminal é um iniciador de si mesmo.

A rotina que determina o conjunto de símbolos iniciadores para cada não-terminal encontra-se na página A16 do apêndice.

Não entraremos em detalhes sobre os algoritmos de fechamento transitivo. Um dos algoritmos eficiente é o descrito por WARSHALL[4]. O algoritmo que usamos é para um caso particu

lar, uma vez que a matriz é esparsa e a metade do lado esquerdo da matriz (a partir da diagonal principal) não é necessária, porque os testes necessitam somente dos terminais que são símbolos iniciadores de cada não-terminal, GRIFFITHS[11].

A rotina que calcula o fechamento transitivo de uma matriz de bits encontra-se na página A16 do apêndice.

IV.2.2 - CONJUNTOS DE SÍMBOLOS SEGUIDORES

O conjunto seguidores de cada não-terminal é também acumulado em uma matriz de bits. A matriz de símbolos seguidores é produzida durante um passo na gramática. Esta matriz necessita de três campos como mostraremos a seguir.

Por exemplo, na regra:

A → B C D | E f

as deduções imediatas são:

- C é seguidor de B
- se C pode gerar a sequência vazia, D é seguidor de B
- D é seguidor de C
- f é seguidor de E

Surge entretanto, um problema. Consideremos uma produção contendo A.

X → Y A Z

Z é seguidor de A. Mas se substituímos A por BCD, obtemos:

X → Y B C D Z

e Z é também seguidor de D. Assim, o fato que D é o último símbolo de A necessita ser indicado, uma vez que todos os seguidores de A são seguidores de D (e se D pode gerar a sequência vazia, o mesmo é verdade para C, e assim por diante).

A matriz de símbolos seguidores tem a forma mostrada na figura IV.4.

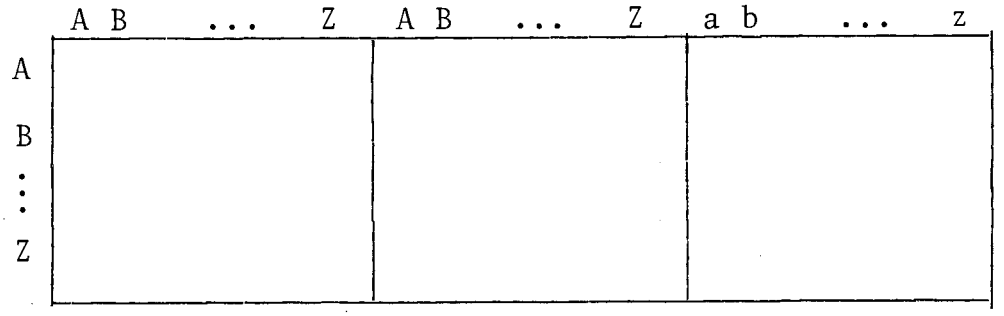

Fig. IV.4

No primeiro campo (X,Y) = 1 indica que X é seguidor de Y.

No segundo campo (X,Y) = 1 indica que os seguidores de X são seguidores de Y, isto é, que Y é o último membro de uma produção de X.

No terceiro campo (x,Y) = 1 indica que x é seguidor de Y.

No primeiro campo, X é seguidor de Y, significa que todos os iniciadores de X são seguidores de Y e estes iniciadores são encontrados na matriz de iniciadores. Uma vez que estamos interessados somente em iniciadores e seguidores terminais podemos adicionar a correspondente linha da matriz de iniciadores ao terceiro campo da matriz de seguidores para cada 1 no primeiro campo.

Efetuando o fechamento transitivo nos segundo e terceiro campos da matriz obtemos a matriz completa de símbolos seguidores.

As rotinas usadas para a determinação do conjunto dos símbolos seguidores de cada não-terminal encontram-se na página A17 do apêndice.

IV.2.3 - CONJUNTOS DE SÍMBOLOS DIRETORES

Depois de obtidas as matrizes de símbolos iniciadores e seguidores, torna-se possível o cálculo das funções I e S, definidas no capítulo anterior, e assim permitir o cálculo dos símbolos diretores e posteriormente a aplicação da condição LL(1).

O algoritmo que segue, foi usado para determinar o conjunto de símbolos diretores de cada não-terminal. A idéia básica do algoritmo é simples:

Para cada produção $A \rightarrow \alpha$, da gramática, são executadas as seguintes ações:

a) para cada terminal a em I (α) introduza $A \rightarrow \alpha$ em $T|A,a|$

b) se A gera a sequência vazia introduza $A \rightarrow \varepsilon$ em $T|A,b|$ para cada terminal b em S(A).

A rotina que determina os conjuntos de símbolos diretores e a que aplica a condição a condição LL(1) encontram-se na pagina A22 do apêndice.

Os conjuntos de símbolos diretores são necessários na geração do analisador sintático, uma vez que eles formam o critério de decisão. Se a condição fornece um resultado positivo, o analisador pode ser gerado diretamente, senão a gramática necessita de modificações antes de ser aceita.

IV.3 - TRANSFORMAÇÃO DE GRAMÁTICAS EM EQUIVALENTES LL(1)

Neste módulo descreveremos as heurísticas usadas na tentativa de reescrever, uma gramática G não LL(1), em uma nova forma, de modo que a nova gramática G' seja LL(1). Este processo no entanto não pode ser inteiramente automático.

Mais formalmente, isto pode ser visto como um problema de decidibilidade. Enquanto é decidível dizer se uma gramática é ou não LL(1), é indecidível dizer se a linguagem descrita por uma gramática livre de contexto arbitrária, é ou não uma linguagem LL(1) e desta forma não é possível escrever um algoritmo de transfromação que funcione em todos os casos.

Por essa razão, não se pode garantir que o programa transformará uma gramática dada em uma equivalente LL(1) mesmo que esta gramática exista. No entanto, esperamos que as heurísticas apresentadas cubram uma larga faixa das gramáticas encontradas na prática.

As principais técnicas usadas para transformação de uma gramática são:

- eliminação de recursão à esquerda
- fatoração e substituição

IV.3.1 - ELIMINAÇÃO DE RECURSÃO À ESQUERDA

Uma das dificuldades que encontramos ao tratar com métodos de análise top-down é a recursão à esquerda.

Uma gramática é recursiva à esquerda se possui um não-terminal A tal que:

$$A \Rightarrow^{+} A\,\alpha$$

para algum α .

O problema de eliminação de recursão à esquerda é resolvido teoricamente em GREIBACH[5] e o algoritmo que usamos é de FOSTER[2]. Este algoritmo utiliza técnicas de representação normalmente associados à álgebra de expressões regulares.

Consideremos um conjunto de não-terminais mutuamente recursivos à esquerda:

$$H = \{X_1, X_2, \ldots X_n\}$$

É possível escrever os membros de H da seguinte forma:

$$X_i \rightarrow X_1\,\beta_{1i} \mid \ldots \mid X_j\,\beta_{ji} \mid \ldots \mid X_n\,\beta_{ni} \mid \alpha_i$$

onde:

$$\alpha_i,\ \beta_{ji} \in (\Sigma \cup N)^*$$

Sem perda de generalidade, podemos reescrever α_i e β_{ji} como não-terminais, isto é, acrescentando as seguintes regras à gramática:

$$A_i \rightarrow \alpha_i$$

$$B_{ji} \rightarrow \beta_{ji}$$

O conjunto de regras pode, ainda, ser reescrito usando os operadores de multiplicação e adição:

$$X_i = X_1 B_{1i} + X_2 B_{2i} + \ldots + X_n B_{ni} + A_i$$

estas equações formam a equação matricial:

$$X = X B + A$$

onde: $X = (X_1 \; X_2 \; \ldots \; X_n)$

$A = (A_1 \; A_2 \; \ldots \; A_n)$

$$B = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1n} \\ B_{21} & B_{22} & \cdots & B_{2n} \\ \vdots & & & \\ B_{n1} & B_{n2} & \cdots & B_{nn} \end{pmatrix}$$

O uso desses operadores pode ser justificado da seguinte forma:

- multiplicação (concatenação) é associativa e a sequência vazia ε serve como elemento identidade.
- adição (alternação) é associativa e comutativa com Φ como elemento identidade.

A matriz identidade é:

$$I = \begin{pmatrix} \varepsilon & \Phi & \Phi & \ldots & \Phi \\ \Phi & \varepsilon & \Phi & & \Phi \\ \vdots & & & & \\ \Phi & \Phi & \Phi & \ldots & \varepsilon \end{pmatrix}$$

com ε na diagonal e Φ nas posições restantes.

A solução mínima da equação matricial $X = X\,B + A$ é:

$$X = A\,B^*$$

onde: $B^* = I + B + B^2 + \ldots$

fazendo $Z = B^*$ e $B^* = I + B\,B^*$, obtemos:

$$Z = I + B\,Z$$

$$X = A\,Z$$

onde Z é a matriz de novos não-terminais:

$$Z = \begin{pmatrix} Z_{11} & Z_{12} & \cdots & Z_{1n} \\ Z_{21} & Z_{22} & \cdots & Z_{2n} \\ \vdots & & & \\ Z_{n1} & Z_{n2} & \cdots & Z_{nn} \end{pmatrix}$$

Vamos ilustrar este método de eliminação de recursão à esquerda com o exemplo a seguir.

Consideremos as produções:

$$A \rightarrow A\ c \mid B\ d \mid e$$

$$B \rightarrow A\ f \mid B\ g \mid h$$

onde os não-terminais A e B são mutuamente recursivos.

Assim: $X = (A \quad B)$

$$Z = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$$

$$A = (e \quad h)$$

$$B = \begin{pmatrix} c & f \\ d & g \end{pmatrix}$$

A solução é:

$$(A \quad B) = (e \quad h) \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$$

$$\begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix} = \begin{pmatrix} \varepsilon & \Phi \\ \Phi & \varepsilon \end{pmatrix} + \begin{pmatrix} c & f \\ d & g \end{pmatrix} \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$$

Isto é:

$A \rightarrow e\ Z_{11} \mid h\ Z_{21}$

$B \rightarrow e\ Z_{12} \mid h\ Z_{22}$

$Z_{11} \rightarrow \varepsilon \mid c\ Z_{11} \mid f\ Z_{21}$

$Z_{21} \rightarrow d\ Z_{11} \mid g\ Z_{21}$

$Z_{12} \rightarrow c\ Z_{12} \mid f\ Z_{22}$

$Z_{22} \rightarrow \varepsilon \mid d\ Z_{12} \mid g\ Z_{22}$

Este método de eliminação de recursão tem a vantagem de não introduzir novos conflitos na condição LL(1).

Como o método é geral, em certos casos a eliminação produz um grande número de novas produções ficando a critério do usuário do sistema usá-lo ou escrever suas gramáticas sem recursão à esquerda.

Algoritmo usado para eliminação de recursão à esquerda:

Seja $H = \{X_1, X_2, \ldots, X_n\}$, o conjunto de não-terminais mutuamente recursivos.

1. Crie as produções $Z_{ii} \rightarrow \varepsilon$ para $i = 1, \ldots, n$

2. Varra as produções de cada X_i

   Se o primeiro símbolo da produção é X_j (o resto é β_{ji})

   - crie produções do tipo:

     $Z_{jk} \rightarrow \beta_{ji} \; Z_{iK}$, para $K = 1, \ldots, n$

   - destrua a produção $X_i \rightarrow X_j \; \beta_{ji}$

   Senão

   - crie produções do tipo:

     $X_K \rightarrow \alpha_i \; Z_{iK}$, para $K = 1, \ldots, n$

   - destrua a produção $X_i \rightarrow \alpha_i$ (atual)

A rotina que elimina recursão à esquerda encontra-se na página A18 do apêndice.

IV.3.2 - FATORAÇÃO

Uma outra técnica que permite transformar gramáticas, não LL(1) em gramáticas equivalente LL(1) é a fatoração.

Por exemplo, se tivermos as produções:

S → do S while C

S → do S until C

S → a

C → b

a presença do símbolo de entrada do não é determinante para a escolha da produção a ser usada pelo analisador.

Um método útil para manipular gramáticas em uma forma conveniente para análise em descida recursivas é fatoração à esquerda - processo de fatorar os prefixos comuns das alternativas.

Se $A \rightarrow \alpha\ \beta \mid \alpha\ \gamma$ são duas produções do não-terminal A, com $\alpha \neq \varepsilon$ e $\beta \neq \gamma$, e a entrada começa com uma sequência não vazia derivada de α, não sabemos se expandimos A para $\alpha\ \beta$ ou $\alpha\ \gamma$. Podemos adiar a decisão expandindo A para $\alpha\ A'$. Então, depois de ver a entrada derivada de α, expandimos A' para β ou para γ. Isto é, após a fatoração as produções originais de A tornam-se:

$A \rightarrow \alpha A'$

$A' \rightarrow \beta \mid \gamma$

Ilustraremos esse processo da seguinte maneira:

Considere a gramática do exemplo anterior, depois de fatorada obtemos:

S → do S S' | a

S' → while C | until C

C → b

Assim podemos expandir S para do S S' para o símbolo de entrada do, e aguardarmos até que do S tenha sido analizado para decidir se expandimos S' para while C ou para until C.

Vejamos agora como a ordem de escolha das produções a serem fatoradas influência na gramática resultante ser ou não LL(1).

Exemplo: Consideremos as produções do não-terminal A:

A → a b c d | (1)

a b f | (2)

a b c e | (3)

a g | (4)

efetuando a fatoração na ordem em que as produções ocorrem obtemos:

A → a D

D → b C | g

C → B | c e

B → c d | f

A gramática resultante não é LL(1) uma vez que os conjuntos de símbolos diretores das expansões do não-terminal C não são disjuntos.

Se efetuarmos a fatoração em outra ordem, por exemplo: produções 1 e 3, o resultado sendo fatorado com a produção 2 e finalmente este com a produção 4, teremos:

A → a D

D → b C | g

C → c B | f

B → d | e

cuja gramática é LL(1).

Com este exemplo podemos afirmar que a fatoração melhora quando efetuada primeiro entre pares de expansões com maior prefixo em comum.

Este princípio é utilizado na rotina que efetua a fatoração na gramática, que se encontra na página A5 do apêndice.

V - EXEMPLO E MODO DE UTILIZAÇÃO

Neste capítulo apresentamos uma gramática exemplo e procuramos fornecer uma interpretação das tabelas geradas.

São forneceidos ao sistema três tipos de cartões. No primeiro cartão o usuário especifica que tipo de operação deve ser efetuada em sua gramática como também alterações do default (mostrado na página 28). Este cartão é dividido em campos, compostos de três colunas consecutivas. No primeiro campo o usuário indica a escolha de uma das alternativas abaixo:

- somente fatorar a gramática (# F)
- somente eliminar recursão à esquerda (# R)
- fatorar e eliminar recursão à esquerda (# FR)
- testar a condição LL(1) para a gramática (# L)
- testar a condição LL(1) modificando a gramática se necessário (# T),

nos campos seguintes, a alteração do default, nesta ordem: número máximo de não-terminais, número máximo de terminais, número de produções média por não-terminal e o comprimento médio do lado direito da produção. No segundo cartão o usuário indica o que deseja que o sistema imprima como resultado parcial, tal como:

- representação interna da gramática
- gramática codificada
- gramática após fatoração

- gramática após eliminação de recursão à esquerda
- tabela de símbolos iniciadores e seguidores,

isto é feito colocando um T, na coluna correspondente, na ordem indicada acima. Os outros cartões são os cartões da gramática

A gramática G, página B1, encontra-se no formato de entrada. A numeração ao lado das produções corresponde ao número do cartão lido.

A gramática G então sofre uma codificação e sua representação interna, página B2, como podemos observar, apresenta a seguinte disposição: na primeira coluna temos o número da produção, na segunda, o código do não-terminal correspondente ao lado esquerdo da produção e da terceira coluna em diante os códigos dos terminais e/ou não-terminais correspondentes ao lado direito da produção.

A gramática codificada encontra-se na página B3. A numeração ao lado das produções tem por objetivo facilitar ao usuário qualquer tipo de modificação que se fizer necessária em uma determinada produção, uma vez que o primeiro número é o do cartão, onde a produção começa, e o segundo número é o da produção. Por exemplo, a produção de número 15, do não-terminal VARIAVEL, foi retirada do cartão de número 24.

Como podemos observar na gramática codificada, as produções dos não-terminais VARIAVEL (14 e 15) e EXPRESSAO (16 e 17) podem ser fatoradas. A gramática fornecida pelo sistema após a fatoração encontra-se na página B5 . As produções 14 e 15 e os não-terminais NT18 e NT19 foram criados pelo sistema.

Na página B7 , temos os não-terminais que são recursi

vos à esquerda e a gramática resultante, após eliminação da recursão acha-se na página B8. Aqui também as produções 3, 4, 15, 24, 32 e os não-teminais NT20, NT21, NT22, NT23 e NT24 foram criados pelo sistema.

Neste ponto, a gramática não sofrerá mais nenhuma transformação e o sistema fornece os códigos do alfabeto da gramática, página B11.

Agora, a condição LL(1) é testada após o que são fornecidas ao usuário as seguintes tabelas:

- conjunto de símbolos iniciadores, página B12.

Nesta tabela, como mostramos em IV.3, temos na ordenada os códigos dos não-terminais e na abscissa os códigos dos não-terminais. Os símbolos terminais iniciadores de cada não-terminal são encontradas da tabela da página B13, e identificados pela presença de um 1. Assim, por exemplo, o conjunto de símbolos iniciadores do não-terminal COMANDO são 'BEGIN', 'GO TO', 'IF', 'RESULT' e 'ID'.

- conjunto de símbolos seguidores, página B14.

Nesta tabela temos na ordenada os códigos dos não-terminais seguidos de outro conjunto de códigos de não-terminais e finalmente os códigos dos terminais. Os símbolos terminais seguidores de cada não-terminal são encontrados na porção da tabela da página B16, e identificados pela presença de um 1. Por exemplo, o conjunto de símbolos seguidores do não-terminal COMANDO são 'END' e ';'.

Tanto na tabela do conjunto de símbolos iniciadores como na tabela do conjunto de símbolos seguidores o interesse

prático está na porção de não-terminais "versus" terminais. A presença de um 1 nesta região significa que o terminal é iniciador do não-terminal correspondente, na primeira tabela, e que o termianl é seguidor do não-terminal correspondente na segunda tabela.

- conjunto de símbolos diretores, página B17.

O formato de saída desse conjunto não é matricial, uma vez que a mesma é esparsa e assim optamos pela saída da seguinte forma: código do não-terminal seguido de pares de valores que representam o código do terminal e o número da produção, respectivamente.

A condição LL(1) para esta gramática não foi satisfeita, como podemos observar na página B18, uma vez que para a entrada |6,48| da tabela de símbolos diretores, isto é, não-terminal COMANDO e terminal 'ID', tivemos duas opções de produções (as de número 6 e 11).

Observando a gramática podemos notar que este conflito pode ser facilmente eliminado se substituirmos as expansões do não-terminal VARIAVEL na produção de número 6 e a expansão do não-termianl LABEL na produção de número 11.

Esta nova gramatica, página B19, foi introduzida no sistema, geradas novas tabelas e testada a condição LL(1).

Como a condição LL(1) foi satisfeita, o analisador sintático pode então ser gerado diretamente.

VI - CONCLUSÕES

Como foi visto no decorrer deste trabalho, a geração de um analisador sintático para uma gramática LL(1), e a própria verificação de que uma gramática livre de contexto arbitrária é ou não LL(1) são tarefas simples, rápidas e eficientes, pelo menos em comparação com outros métodos de construção de analisadores sintáticos.

A tese foi utilizada como instrumento didático em disciplinas de compiladores em curso de graduação do Instituto de Matemática da UFRJ, bem como na construção do analisador sintático top-down para a linguagem de simulação de sistemas - SIMULA, em um projeto desenvolvido no Departamento de Computação e Estatística da Universidade Federal de São Carlos - SP, que teve como objetivo a construção e comparação de dois analisadores sintáticos para a referida linguagem usando os métodos LL(1) e precedência de operadores.

Este gerador constitui uma das partes de um projeto desenvolvido pela linha de pesquisa de Compiladores do Programa de Engenharia de Sistemas e Computação da COPPE, para formação de um laboratório de compiladores e técnicas de compilação.

No que toca a análise sintática LL(1), esse projeto deve incluir também como extensão deste trabalho, a construção de um gerador automático de analisadores sintáticos com recuperação de erro, estando também prevista sua adaptação para funcionamento on-line, através de terminais.

BIBLIOGRAFIA

|1| GRIFFITHS, M - LL(1) Grammars and Analysers, France, Laboratoire d' Informatique Universite de Grenoble. Compiler Construction: An Advanced Course, Bauer & Eickel cap 2.B, 57 - 82.

|2| FOSTER, J.M - A Syntax Improving Device. Computer Journal, maio de 1968.

|3| KNUTH, Donald E. - Top-Down Syntax Analysis. Acta Informatica, vol 1, 79 - 110, 1971.

|4| WARSHALL, S. - A Theorem on Boolean Matrices. JACM, vol 9, nº 1, janeiro de 1962

|5| GREIBACH, S.A - A New Normal Form Theorem for Context - Free Phrase Structure Grammars. JACM, 12, 42-52, 1965

|6| HUNT III, H.B., SZYMANSKI, T.G., ULLMAN, J.D. - Operations on Sparse Relations. Communications of the ACM, vol. 20, nº 3, março de 1977.

|7| AHO, Alfred V., ULLMAN, Jefferey D. - Basic Parsing Techniques. Principles of Compiler Design. Addison-Wesley, 1977.

|8| AHO, Alfred V., ULLMAN, Jeffrey D. - One-Pass No Backtrack Parsing. The Theory of Parsing, Translation and Compiling, vol I: Parsing. Prentice-Hall, 1972.

|9| AHO, Alfred V., ULLMAN, Jeffrey D. - Theory of Deterministic Parsing. The Theory of Parsing, Translation and Compiling, vol II: Compiling. Prentice-Hall, 1973.

|10| GRIES, David - Construction the Transitive Closure of Relations. Compiler Construction for Digital Computers John Wiley & Sons, 1971.

|11| GRIFFITHS, M. - Analyse Deterministe et Compilateurs - Tese, University of Grenoble, outubro de 1969.

```
BEGIN
  FILE INPUT(KIND=READER),
       OUTPUT(KIND=PRINTER);
  INTEGER MAXNTER,MAXTER,PRODNTER,COMPROD,N1,N2,N3,INDICADOR,POOL;
  READ(INPUT,<A3,4I3>,INDICADOR,MAXNTER,MAXTER,PRODNTER,COMPROD);
  COMMENT
   MAXNTER - NUMERO MAXIMO DE NAO-TERMINAIS
   MAXTER  - NUMERO MAXIMO DE TERMINAIS
   PRODNTER- PRODUCAO MEDIA POR NAO-TERMINAL
   COMPROD - COMPRIMENTO MEDIO DO LADO DIREITO DA PRODUCAO
   POOL    - NUMERO DE BUFFER DISPONIVEIS;
  IF   MAXNTER = 0
  THEN BEGIN
         MAXNTER :=200;
         MAXTER  :=500;
         PRODNTER:=4;
         COMPROD :=5;
       END;
  POOL:=MAXNTER*COMPROD*PRODNTER;
  N3:= MAXTER DIV 48 + 1;
  N1:= (MAXNTER+MAXTER) DIV 48 + 1;
  N2:= (2*MAXNTER+MAXTER) DIV 48 + 1;
  BEGIN
    ALPHA ARRAY   TABSIMB[1:2*(MAXNTER+MAXTER)];
    INTEGER ARRAY NO[1:POOL],
                  PONT[1:MAXNTER],
                  PINTER[1:MAXNTER],
                  V[1:MAXNTER],
                  Z[1:48*N2],
                  SREAL[1:MAXNTER,1:N1],
                  FREAL[1:MAXNTER,1:N2],
                  DS[1:MAXNTER],
                  LADODIR[1:3,1:N3],
                  VETSIMB[1:MAXNTER+MAXTER],
                  VNT[1:MAXNTER],
                  NODO[1:10*MAXNTER];
    COMMENT
     NO     - MEMORIA DISPONIVEL PARA REPRESENTACAO INTERNA DA GRAMATICA
     PONT   - PONTEIRO PARA O NO' INICIAL DA PRIMEIRA PRODUCAO DE CADA
              NAO-TERMINAL
     PINTER- VETOR QUE CONTEM O NUMERO DE PRODUCOES DE CADA NAO-TERMINAL
              E O PONTEIRO PARA O NO' INICIAL DE SUA PRIMEIRA PRODUCAO
     V      - VETOR QUE INDICA SE UM NAO-TERMINAL GERA OU NAO A SEQUENCIA
              VAZIA
     SREAL  - MATRIZ QUE CONTEM O CONJUNTO DE SIMBOLOS STARTER
     FREAL  - MATRIZ QUE CONTEM O CONJUNTO DE SIMBOLOS FOLLOWER
     DS     - PONTEIRO PARA A LISTA DE SIMBOLOS DIRETORES DE CADA
              NAO-TERMINAL DA GRAMATICA
     VNT    - VETOR DE ORDEM DOS SIMBOLOS NAO-TERMINAIS PARA O CALCULO
              DAS FUNCOES START, FOLLOW E SIMBOLOS DIRETORES
     NODO   - MEMORIA DISPONIVEL PARA A GERACAO DA MATRIZ ESPARSA DE
              SIMBOLOS DIRETORES
     LADODIR - MATRIZ GERADORA DOS SIMBOLOS DIRETORES DE CADA NAO-
              TERMINAL
     VETSIMB - VETOR QUE CONTEM O NOME DOS SIMBOLOS NAO-TERMINAIS E
              TERMINAIS DA GRAMATICA ORIGINAL
     TABSIMB - TABELA DE SIMBOLOS DA GRAMATICA ;
    INTEGER I,J,K,NTER,TER,LIMNTER,TOTNTER,DISP,CANAL;
    BOOLEAN RECURSAO,CHEGANT,POOL,FLAG,FAT,REC,LL1,TR,RI,GC,GR,GF,TAB;
    ALPHA PALAVRA;
    DEFINE LLINK(I)   = NO[I].[47:16] #,
           CODIGO(I)  = NO[I].[31:16] #,
           RLINK(I)   = NO[I].[15:16] #,
           PTR(I)     = RLINK(I) #,
           NCARD(I)   = CODIGO(I) #,
           INP(I)     = PINTER[I].[47:24] #,
           PINI(I)    = PINTER[I].[23:24] #,
           XR(X,I,J)  = X[I,(J-1) DIV 48 + 1].[(J-1) MOD 48 : 1] #,
           S(I,J)     = XR(SREAL,I,J) #,
           F(I,J)     = XR(FREAL,I,J) #,
           CODTER(I)  = NODO[I].[47:16] #,
           NPROD(I)   = NODO[I].[31:16] #,
           LINK(I)    = NODO[I].[15:16] #,
           DIR(I,J)   = XR(LADODIR,I,J) #,
           INICIO(I)  = VETSIMB[I].[47:24] #,
           COMP(I)    = VETSIMB[I].[23:24] #;
```

```
{****}
{  PROCEDIMENTO QUE CRIA UM NOME PARA UM NOVO NAO-TERMINAL                }
{****}
PROCEDURE NOVONAOTERMINAL(X);
INTEGER X;
BEGIN
  INTEGER A,B,C;
  DEFINE NOME(A,B) = PALAVRA.[B:8]:=A+240 #;
  PALAVRA:="N(      ";
  C:=X DIV 100;
  IF   C <> 0
  THEN BEGIN
         C:=X DIV 10;
         NOME(C DIV 10,31);
         NOME(C MOD 10,23);
         NOME(X MOD 10,15);
       END
  ELSE BEGIN
         NOME(X DIV 10,31);
         NOME(X MOD 10,23)
       END
END;
{****}
{  PROCEDIMENTO QUE IMPRIME O ALFABETO DA GRAMATICA                       }
{****}
PROCEDURE ALFABETO;
BEGIN
  INTEGER I,J;
  WRITE(OUTPUT[SKIP 1]);
  WRITE(OUTPUT,<//,X4," CODIGO DO ALFABETO",//>);
  FOR I:=1 STEP 1 UNTIL NTER DO
  WRITE(OUTPUT,<X6,I4,X4,"A6>,I,COMP(I),FOR J:=1 STEP 1 UNTIL
        COMP(I) DO TABSIMB[INICIO(I)+J-1]);
  FOR I:=NTER+1 STEP 1 UNTIL LIMNTER DO
  BEGIN
    NOVONAOTERMINAL(I);
    WRITE(OUTPUT,<X6,I4,X4,A6>,I,PALAVRA)
  END;
  FOR I:=MAXNTER+1 STEP 1 UNTIL MAXNTER+TER DO
  WRITE(OUTPUT,<X6,I4,X4,"A6>,I-MAXNTER+LIMNTER,COMP(I),FOR J:=1
        STEP 1 UNTIL COMP(I) DO TABSIMB[INICIO(I)+J-1]);
END;
{****}
{  PROCEDIMENTO QUE IMPRIME A REPRESENTACAO INTERNA DA GRAMATICA          }
{****}
PROCEDURE GRAMATICACODIFICADA;
BEGIN
  INTEGER ARRAY S[1:23];
  INTEGER I,J,K,X,Y,NP;
  WRITE(OUTPUT[SKIP 1]);
  WRITE(OUTPUT,<X25,80("*"),/,X25,"*",T105,"*",/,X25,"*",X22,
        "REPRESENTACAO INTERNA DA GRAMATICA",T105,"*",/,X25,"*",
        T105,"*",/,X25,80("*"),//>);
  WRITE(OUTPUT,<///," NUMERO DA LADO     LADO",/," PRODUCAO  ESQ",
        "UERDO DIREITO",//>);
  FOR I:=1 STEP 1 UNTIL NTER DO
  BEGIN
    X:=PONTI[I];
    WHILE X <> 0 DO
    BEGIN
      K:=0;
      NP:=NP+1;
      Y:=PTR(X);
      WHILE Y <> 0 DO
      BEGIN
        K:=K+1;
        S[K]:=CODIGO(Y);
        Y:=RLINK(Y);
      END;
      WRITE(OUTPUT,<I6,X4,I5,X4,22I5>,NP,I,FOR J:=1 STEP 1
            UNTIL K DO S[J]);
      X:=LLINK(X);
    END
  END;
  WRITE(OUTPUT,<2(/)," NUMERO DE NAO-TERMINAIS =",I5,/," NUMER",
        "O DE TERMINAIS =",I5>,NTER,TER);
END;
```

```
{***
    PROCEDIMENTO QUE IMPRIME A GRAMATICA APOS ALGUMA TRANSFORMACAO
***}

PROCEDURE GRAMATICAFINAL(Z);
INTEGER Z;
BEGIN
  ALPHA ARRAY BUFFER[1:13,PROINTER];
  INTEGER I,J,K,L,X,Y,CONT,LINHA;
  PROCEDURE TITULO;
  BEGIN
    WRITE(OUTPUT[SKIP 1]);
    WRITE(OUTPUT,<X25,80("*"),/,X25,"*",T105,"*">);
    CASE Z OF
    BEGIN
    1:WRITE(OUTPUT,<X25,"*",X25,"GRAMATICA CODIFICADA",T105,"*">);
    2:WRITE(OUTPUT,<X25,"*",X25,"GRAMATICA APOS FATORACAO",T105,
        "*">);
    3:WRITE(OUTPUT,<X25,"*",X25,"GRAMATICA APOS ELIMINACAO DA ",
        "RECURSAO",T105,"*">);
    END;
    WRITE(OUTPUT,<X25,"*",T105,"*",/,X25,80("*"),///>);
  END;
  TITULO;
  FOR I:=1 STEP 1 UNTIL LIMNTER DO
  BEGIN
    L:=0;
    IF   I <= NTER
    THEN FOR J:=1 STEP 1 UNTIL COMP(I) DO
         BEGIN
           L:=++L;
           BUFFER[L]:=TABSIMB[INICIO(I)+J-1]
         END
    ELSE BEGIN
           NOVONAOTERMINAL(I);
           L:=++1;
           BUFFER[L]:=PALAVRA
         END;
    L:=++1;
    BUFFER[L]:="  =   ";
    X:=PONT[I];
    WHILE X <> 0 DO
    BEGIN
      CONT:=++1;
      IF   PTR(X) = 0
      THEN CASE Z OF
           BEGIN
             1:WRITE(OUTPUT,<2I5,X2,*A6,/>,NCARD(X),CONT,L,FOR J:=1
               STEP 1 UNTIL L DO BUFFER[J]);
           2:3:WRITE(OUTPUT,< I7,X5,*A6,/>,CONT,L,FOR J:=1 STEP 1
               UNTIL L DO BUFFER[J]);
           END
      ELSE BEGIN
             K:=L;
             Y:=PTR(X);
             WHILE Y <> 0 DO
             BEGIN
               IF   CODIGO(Y) > NTER AND CODIGO(Y) <= MAXNTER
               THEN BEGIN
                      NOVONAOTERMINAL(CODIGO(Y));
                      K:=++1;
                      BUFFER[K]:=PALAVRA
                    END
               ELSE FOR J:=1 STEP 1 UNTIL COMP(CODIGO(Y)) DO
                    BEGIN
                      K:=++1;
                      BUFFER[K]:=TABSIMB[INICIO(CODIGO(Y))+J-1]
                    END;
               K:=++1;
               BUFFER[K]:="  .  ";
               Y:=RLINK(Y)
             END;
             IF K <= 20 THEN Y:=K ELSE BEGIN
                                         Y:=20;
                                         LINHA:=++1;
                                       END;
```

```
                CASE Z OF
                BEGIN
                  1:WRITE(OUTPUT,<  2I5,X2,20A6>,NCARD(X),CONT,FOR
                          J:=1 STEP 1 UNTIL Y DO BUFFER[J]);
                  2:3:WRITE(OUTPUT,<   I7,X5,20A6>,CONT,FOR J:=1 STEP 1
                          UNTIL Y DO BUFFER[J]);
                END;
                Y:=132-(6*L+12) DIV 6;
                WRITE(OUTPUT,<  X12,X*,*A6>,6*L,Y,FOR J:=21 STEP 1
                      UNTIL K DO BUFFER[J]);
              END;
          FOR J:=1 STEP 1 UNTIL L-1 DO BUFFER[J]:="      ";
          BUFFER[L]:="  |   ";
          LINHA:=*+1;
          IF LINHA >= 36 THEN BEGIN
                                TITULO;
                                LINHA:=3;
                              END;
          X:=LLINK(X)
        END;
      END;
      CASE Z OF
      BEGIN
       1:WRITE(OUTPUT,<//,"  OBS.: A NUMERACAO AO LADO DAS PRODUCOES ",
              "CORRESPONDE",/,X7,"AO NUMERO DA CARTAO ONDE A PRODU",
              "CAO INICIA(PRIMEIRO NUMERO) E AO NUMERO DA PRODUCAO ",
              "(SEGUNDO NUMERO)",/>);
       2:3:WRITE(OUTPUT,<//,"  OBS.: A NUMERACAO AO LADO DAS PRODUCOE",
              "S CORRESPONDE AO NUMERO DA PRODUCAO",/>);
      END;
    END;
%                                                                      %
%   PROCEDIMENTO QUE IMPRIME A ESTRUTURA LINKADA DA GRAMATICA          %
%                                                                      %
    PROCEDURE DUMP(LI,LS);
    INTEGER LI,LS;
    BEGIN
      INTEGER ARRAY A[0:30];
      INTEGER I,J,K,L,X;
      WRITE(OUTPUT[SKIP 1]);
      WRITE(OUTPUT,<//,"  PONT  ESQUERDA DIREITA",//>);
      FOR I:=LI STEP 1 UNTIL LS DO
      BEGIN
        J:=PONT[I];
        A[Y]:=J;
        WHILE J <> 0 DO
        BEGIN
          A[X+1]:=LLINK(J);
          A[X+2]:=RLINK(J);
          X:=X+2;
          K:=PTR(J);
          WHILE K <> 0 DO
          BEGIN
            A[X+1]:=CODIGO(K);
            A[X+2]:=RLINK(K);
            X:=X+2;
            K:=RLINK(K);
          END;
          IF X > 24 THEN K:=24 ELSE K:=X;
          WRITE(OUTPUT,</,I5,X1,24I5>,A[0],FOR L:=1 STEP 1 UNTIL K DO
                A[L]);
          WRITE(OUTPUT,<X6,24I5>,FOR L:=25 STEP 1 UNTIL X DO A[L]);
          X:=0;
          J:=LLINK(J)
        END
      END
    END;
```

```
%%%%  PROCEDIMENTO QUE DETETA E EFETUA A FATORACAO NA GRAMATICA  %%%%

  PROCEDURE DETETAFATORABILIDADE;
  BEGIN
   DEFINE MAXPROD = 30 #;
   BEGIN
    BOOLEAN FATORE;  LABEL FINAL;
    INTEGER ARRAY VETOR[1:MAXPROD],
                  MATRIZ[1:MAXPROD,1:MAXPROD];
    INTEGER NPR,IFN,NSC,AES,ORD,BACK,K1,K2,K3,PT1,PT2;
    DEFINE V1(I) = VETOR[I].[47:24] #,
           V2(I) = VETOR[I].[23:24] #;
    %  NPR  : NUMERO DE PRODUCOES POR NAO-TERMINAL
    %  IFN  : INDICADOR DE FALTA DE NO DISPONIVEL
    %  NSC  : NUMERO DE SIMBOLOS COMUNS ENTRE DUAS PRODUCOES
    PROCEDURE RECUPERENO(X,Y);
    INTEGER X,Y;
    COMMENT - PROCEDIMENTO QUE DEVOLVE UM NO' A LISTA DE NOS
              DISPONIVEIS;
    BEGIN
      INTEGER A;
      A:=DISP;
      DISP:=X;
      RLINK(Y):=A
    END;
%=======================================================================%
    PROCEDURE GEREPRODUCAO(X,Y);
    INTEGER X,Y;
    COMMENT - PROCEDIMENTO QUE GERA UMA NOVA PRODUCAO PARA O NAO -
              TERMINAL;
    BEGIN
      IFN:=0;
      IF   PTR(DISP) ¬= 0
      THEN BEGIN
             X:=DISP;
             DISP:=PTR(X);
             LLINK(X):=0;
             IF RLINK(Y) ¬= 0 THEN PTR(X):=RLINK(Y)
                              ELSE PTR(X):=0
           END
      ELSE BEGIN
             IFN:=1;
             WRITE(OUTPUT,<///,X10,"ACABOU MEMORIA DISPONIVEL">)
           END
    END;
%=======================================================================%
    PROCEDURE GERMATRIZ(A);
    INTEGER A;
    COMMENT - PROCEDIMENTO QUE GERA A MATRIZ DE ORDENACAO PARA A
              FATORACAO;
    BEGIN
      INTEGER L,C,X,Y;
      X:=PONT[A];
      FOR L:=1 STEP 1 UNTIL NPR DO
      BEGIN
        V1(L):=L;
        V2(L):=X;
        X:=LLINK(X)
      END;
      FOR L:=1 STEP 1 UNTIL NPR-1 DO
        IF PTR(V2(L)) ¬= 0 THEN
        FOR C:=L+1 STEP 1 UNTIL NPR DO
        BEGIN
          INTEGER IGUAL;  BOOLEAN SAI;
          X:=PTR(V2(L));
          Y:=PTR(V2(C));
          WHILE X ¬= 0 AND Y ¬= 0 AND ¬ SAI DO
            IF   CODIGO(X) = CODIGO(Y)
            THEN BEGIN
                   IGUAL:=+1;
                   X:=RLINK(X);
                   Y:=RLINK(Y)
                 END
            ELSE SAI:=TRUE;
          MATRIZ[L,C]:=IGUAL
        END;
    END;
```

```
    PROCEDURE CALCULENPR(A);
    INTEGER A;
    COMMENT - PROCEDIMENTO QUE CALCULA O NUMERO DE PRODUCOES DE UM
              NAO-TERMINAL;
    BEGIN
      INTEGER X;
      X:=PONT[A];
      NPR:=0;
      WHILE X ¬= 0 DO
      BEGIN
        NPR:=+1;
        X:=LLINK(X)
      END
    END;
%=====================================================================%
    PROCEDURE VERSEFATORA(A);
    INTEGER A;
    COMMENT - PROCEDIMENTO QUE VERIFICA SE UM NAO-TERMINAL NECESSITA
              DE FATORACAO;
    BEGIN
      INTEGER ARRAY AQUARELA[1:MAXPROD];
      INTEGER I,J,X;
      LABEL FIM;
      X:=PONT[A];
      IF   PTR(X) ¬= 0
      THEN AQUARELA[1]:=CODIGO(PTR(X));
      FOR I:=2 STEP 1 UNTIL NPR DO
      BEGIN
        X:=LLINK(X);
        IF PTR(X) ¬= 0 THEN
        BEGIN
          FOR J:=1 STEP 1 UNTIL I-1 DO
            IF  CODIGO(PTR(X)) = AQUARELA[J]
            THEN BEGIN
                   FATORE:=TRUE;
                   GO FIM
                 END;
          AQUARELA[I]:=CODIGO(PTR(X))
        END
      END;
      FIM:
    END;
%=====================================================================%
    PROCEDURE DUPLAAFATORAR(X,Y,Z);
    INTEGER X,Y,Z;
    COMMENT - PROCEDIMENTO QUE DETERMINA O PAR DE PRODUCOES A SEREM
              FATORADAS;
    BEGIN
      INTEGER I,J;
      X:=1;
      Y:=2;
      Z:=MATRIZ[1,2];
      FOR I:=1 STEP 1 UNTIL NPR-1 DO
        FOR J:=I+1 STEP 1 UNTIL NPR DO
          IF   Z < MATRIZ[I,J]
          THEN BEGIN
                 Z:=MATRIZ[I,J];
                 X:=I;
                 Y:=J
               END;
      IF   Y-X > 1
      THEN BEGIN
             LABEL FIM;
             FOR I:=Y-1 STEP -1 UNTIL X+1 DO
               IF   V1[I] ¬= 0
               THEN BEGIN
                      BACK:=V2[I];
                      GO FIM
                    END;
             BACK:=0;
             FIM:
           END
      ELSE BACK:=V2[X];
      V1[Y]:=0;
      FOR I:=Y+1 STEP 1 UNTIL NPR DO MATRIZ[Y,I]:=0;
      FOR I:=1 STEP 1 UNTIL Y-1 DO MATRIZ[I,Y]:=0;
END;
```

```
  PROCEDURE GERENTER;
  COMMENT - PROCEDIMENTO QUE GERA UM NOVO NAO-TERMINAL PARA A
            GRAMATICA;
  BEGIN
    INTEGER X,Y;
    LABEL FIM;
    IF   LIMNTER+1 > MAXNTER
    THEN BEGIN
           WRITE(OUTPUT,<///,X10,"O NUMERO MAXIMO DE NAO-TERMINAIS",
                        " JA' FOI ALCANSADO",//>);
           CHEGANT:=TRUE;
           IFN:=1
         END;
    GEREPRODUCAO(X,PT1);
    IF IFN = 1 THEN GO FIM;
    GEREPRODUCAO(Y,PT2);
    IF IFN = 1 THEN GO FIM;
    LLINK(X):=Y;
    LIMNTER:=+1;
    PONT[LIMNTER]:=X;
    VNT[LIMNTER]:=LIMNTER;
    IF BACK = 0 THEN LLINK(V2(ABS)):=LLINK(V2(ORD))
                ELSE LLINK(BACK):=LLINK(V2(ORD));
    RECUPERENO(V2(ORD),PT2);
    IFN:=0;
    IF   RLINK(DISP) <> 0
    THEN BEGIN
           Y:=DISP;
           DISP:=RLINK(Y);
           RLINK(Y):=0;
           CODIGO(Y):=LIMNTER;
           RLINK(PT1):=Y
         END
    ELSE BEGIN
           IFN:=1;
           WRITE(OUTPUT,<///,X10,"ACABOU MEMORIA DISPONIVEL">)
         END;
    FIM:
  END;
%=============================================================%
  FOR K1:=1 STEP 1 UNTIL LIMNTER DO
  BEGIN
    CALCULENPR(K1);
    VERSEFATORA(K1);
    IF FATORE THEN
    BEGIN
      LABEL SAI;
      GEREMATRIZ(K1);
      FOR K2:=1 STEP 1 UNTIL NPR DO
      BEGIN
        DUPLAAFATORAR(ABS,ORD,NSC);
        IF NSC = 0 THEN GO SAI;
        PT1:=PTR(V2(ABS));
        PT2:=PTR(V2(ORD));
        FOR K3:=2 STEP 1 UNTIL NSC DO
        BEGIN
          PT1:=RLINK(PT1);
          PT2:=RLINK(PT2)
        END;
        GERENTER;
        IF IFN = 1 THEN GO FINAL
      END;
      SAI:
      FATORE:=FALSE;
    END
  END;
  IF   PT1 = 0
  THEN WRITE(OUTPUT,<///,10("*"),X10,"A GRAMATICA NAO NECESSITA ",
             "DE FATORACAO",X10,10("*")>)
  ELSE IF   GF THEN GRAMATICAFINAL(2)
       ELSE WRITE(OUTPUT,<///,10("*"),X10,"IMPRESSAO DA GRAMATICA",
                  " FATORADA SUSPENSA PELO USUARIO",X10,10("*")>);
  FINAL:
 END;
END;
```

```
%
%%  PROCEDIMENTO QUE VERIFICA SE UM NAO-TERMINAL PODE GERAR UMA      %%
%%   SEQUENCIA VAZIA                                                 %%
%
    PROCEDURE SEQUENCIAVAZIA;
    BEGIN
      INTEGER ARRAY VPE[1:5*MAXNTER],
                    VETREC[1:PRODNTER*MAXNTER],
                    NOCOPIA[1:30*MAXNTER];
      INTEGER I,J,X,Y,CONT,LOOP,AVAIL;
      BOOLEAN ACHOU;
      LABEL FIM;
      DEFINE PI(I)   = VETREC[I].[47:24] #,
             PF(I)   = VETREC[I].[23:24] #,
             VER(I)  = NOCOPIA[I].[47:16] #,
             HOR(I)  = NOCOPIA[I].[15:16] #,
             SIMB(I) = NOCOPIA[I].[31:16] #;
%=============================================================%
      PROCEDURE COPIA;
      COMMENT - PROCEDIMENTO QUE EFETUA UMA COPIA DA GRAMATICA;
      BEGIN
        INTEGER ARRAY APONTADOR[1:LIMNTER];
        INTEGER I,J,X,Y,Z1,Z2,Z3;
        BOOLEAN BOOL;
        LABEL ERRO;
        PROCEDURE LIBERENO(A);
        INTEGER A;
        BEGIN
          IF   HOR(AVAIL) ¬= 0
          THEN BEGIN
                 A:=AVAIL;
                 AVAIL:=HOR(A)
               END
          ELSE A:=0
        END;
        FOR I:=1 STEP 1 UNTIL LIMNTER DO
        BEGIN
        Y:=PONTI[I];
        LIBERENO(Z1);
        IF Z1 ¬= 0 THEN BEGIN
                          INP(I):=J+1;
                          PINT(I):=Z1
                        END
                   ELSE BEGIN
                          BOOL:=TRUE;
                          GO ERRO
                        END;
        WHILE X ¬= 0 DO
        BEGIN
          J:=J+1;
          PI(J):=Z1;
          IF   LLINK(X) = 0
          THEN VER(Z1):=0
          ELSE BEGIN
                 LIBERENO(Z3);
                 IF Z3 = 0 THEN BEGIN
                                  BOOL:=TRUE;
                                  GO ERRO
                                END
                           ELSE VER(Z1):=Z3
               END;
          IF   PTR(X) = 0
          THEN BEGIN
                 HOR(Z1):=0;
                 Z1:=VER(Z1);
                 PF(J):=0
               END
          ELSE BEGIN
                 LIBERENO(Z2);
                 IF Z2 = 0 THEN BEGIN
                                  BOOL:=TRUE;
                                  GO ERRO
                                END
                           ELSE HOR(Z1):=Z2;
                 Y:=PTR(X);
```

```
                WHILE Y <> 0 DO
                BEGIN
                  LABEL F1;
                  SIMB(Z2):=CODIGO(Y);
                  IF   RLINK(Y) <> 0
                  THEN BEGIN
                         LIBERENO(Z3);
                         IF Z3 = 0 THEN BEGIN
                                          BOOL:=TRUE;
                                          GO ERRO
                                        END
                                   ELSE HOR(Z2):=Z3
                       END
                  ELSE BEGIN
                         HOR(Z2):=0;
                         PP(J):=Z2;
                         Z1:=VER(Z1);
                         GO F1
                       END;
                  Z2:=Z3;
                  F1:
                  Y:=RLINK(Y)
                END
              END;
        X:=LLINK(X)
    END;
    END;
    INP(LIMNTER+1):=J+1;
    FOR I:=1 STEP 1 UNTIL LIMNTER DO APONTADOR[I]:=PINT(I);
    ERRO:
    IF BOOL THEN
    WRITE(OUTPUT,<///,X10,"ACABOU MEMORIA DISPONIVEL">);
  END;
%=============================================================%
  PROCEDURE ELIMINEPROD(NP,NT);
  INTEGER NP,NT;
  COMMENT - PROCEDIMENTO QUE ELIMINA UMA PRODUCAO DE UM NAO -
            TERMINAL;
  BEGIN
    INTEGER I;
    IF   INP(NT+1)-1 = INP(NT)
    THEN PINT(NT):=0
    ELSE BEGIN
           LABEL FIM;
           FOR I:=NP-1 STEP -1 UNTIL INP(NT) DO
             IF   VER[I] <> 1
             THEN BEGIN
                    VER(PI(I)):=VER(PI(NP));
                    GO FIM
                  END;
           PINT(NT):=VER(PI(NP));
           FIM:
         END;
    VPE(NP):=1
  END;
%=============================================================%
  PROCEDURE ELIMINEPRODUCOES(A);
  INTEGER A;
  COMMENT - PROCEDIMENTO QUE ELIMINA TODAS AS PRODUCOES DE UM
            NAO-TERMINAL;
  BEGIN
    INTEGER I;
    FOR I:=INP(A) STEP 1 UNTIL INP(A+1)-1 DO VPE[I]:=1;
    V[A]:=1
  END;
%=============================================================%
  PROCEDURE ELIMINENTER(A);
  INTEGER A;
  COMMENT - PROCEDIMENTO QUE ELIMINA UM NAO-TERMINAL DA COPIA DA
            GRAMATICA;
  BEGIN
    INTEGER I;
    LABEL FIM;
    FOR I:=INP(A) STEP 1 UNTIL INP(A+1)-1 DO
      IF VPE[I] <> 1 THEN GO FIM;
    V[A]:=0;
    FIM:
  END;
```

```
PROCEDURE ELIMINESIMB(A,B,C);
INTEGER A,B,C;
COMMENT - PROCEDIMENTO QUE ELIMINA UM SIMBOLO DE UMA PRODUCAO;
BEGIN
  IF   HOR(B) ¬= 0
  THEN HOR(A):=HOR(B)
  ELSE BEGIN
         HOR(A):=0;
         PF(C):=A
       END;
END;
COMMENT
 O ALGORITMO QUE DETERMINA SE UM NAO-TERMINAL PODE OU NAO GERAR A
 SEQUENCIA VAZIA REQUER UMA COPIA DA GRAMATICA E UM VETOR "V" COM
 UMA ENTRADA POR NAO-TERMINAL. CADA ELEMENTO DE V E' INICIALIZADO
 COM -1 (INDEFINIDO) ;
FOR I:=1 STEP 1 UNTIL LIMNTER DO V(I):=-1;
AVAIL:=1;
FOR I:=1 STEP 1 UNTIL 30*LIMNTER DO HOR(I):=I+1;
HOR(30*LIMNTER):=0;
COPIA;
COMMENT
 DURANTE UM PASSO DA GRAMATICA SAO EXECUTADAS AS SEGUINTES ACOES:
 A- SE QUALQUER EXPANSAO DE UM NAO-TERMINAL E' A SEQUENCIA VAZIA,
    O CORRESPONDENTE ELEMENTO DE V RECEBE O VALOR 1 (SIM) E TODAS
    AS PRODUCOES DO NAO-TERMINAL SAO ELIMINADAS DA GRAMATICA
 B- QUALQUER PRODUCAO CONTENDO UM SIMBOLO TERMINAL E' ELIMINADA
    DA GRAMATICA. SE ESSA ACAO ELIMINA TODAS AS PRODUCOES DO NAO-
    TERMINAL, O VALOR CORRESPONDENTE DE V RECEBE O VALOR 0 (NAO);
FOR I:=1 STEP 1 UNTIL LIMNTER DO
IF PINT(I) ¬= 0 THEN
BEGIN
  LABEL L1;
  FOR J:=INP(I) STEP 1 UNTIL INP(I+1)-1 DO
  IF   PF(J) = 0
  THEN BEGIN
         ELIMINEPRODUCOES(I);
         PINT(I):=0;
         GO L1
       END
  ELSE BEGIN
         X:=HOR(PI(J));
         WHILE X ¬= 0 DO
         IF   SIMB(X) > MAXNTER
         THEN BEGIN
                ELIMINEPROD(J,I);
                X:=0
              END
         ELSE X:=HOR(X)
       END;
  ELIMINENTER(I);
  L1:
END;
COMMENT
 A GRAMATICA AGORA ESTA' LIMITADA PELA REGRA NA QUAL O LADO DI -
 REITO CONTEM SOMENTE SIMBOLOS NAO-TERMINAIS. SUCESSIVOS PASSOS
 NA GRAMATICA SAO EXECUTADAS AS SEGUINTES ACOES :
 A- SE A ENTRADA CORRESPONDENTE DE V TEM O VALOR 1, O SIMBOLO E'
    ELIMINADO. SE ISSO LEVA A SEQUENCIA VAZIA COMO LADO DIREITO. O
    NAO-TERMINAL PARA O QUAL ESTA E' UMA EXPANSAO PODE GERAR A SE-
    QUENCIA VAZIA. A CORRESPONDENTE ENTRADA DE V TORNA-SE 1 E AS
    PRODUCOES DO NAO-TERMINAL SAO ELIMINADAS
 B- SE A CORRESPONDENTE ENTRADA DE V TEM O VALOR 0, A PRODUCAO E'
    ELIMINADA. SE TODAS AS PRODUCOES DO NAO-TERMINAL SAO ELIMINA-
    DAS DESSA MANEIRA, SUA ENTRADA EM V RECEBE O VALOR 0 ;

FOR I:=1 STEP 1 UNTIL LIMNTER DO
  IF V(I) = -1 THEN LOOP:=+1;
WHILE CONT < LOOP DO
BEGIN
  FOR I:=1 STEP 1 UNTIL LIMNTER DO
  BEGIN
    LABEL LNT;
    IF V(I) = -1 THEN
    FOR J:=INP(I) STEP 1 UNTIL INP(I+1)-1 DO
      IF VPE(J) ¬= 1 THEN
```

```
      BEGIN
        X:=PI(J);
        Y:=HOR(X);
        WHILE Y <> 0 DO
        BEGIN
          IF   VISIMB(Y)] <> -1
          THEN IF   V[SIMB(Y)] = 1
               THEN BEGIN
                      ELIMINESIMB(X,Y,J);
                      IF   PF(J) = 0
                      THEN BEGIN
                             ELIMINEPRODUCOES(I);
                             GO LNT
                           END;
                      IF HOR(X) = 0 THEN Y:=X
                    END
               ELSE BEGIN
                      ELIMINEPROD(J,I);
                      ELIMINENTER(I);
                      GO LNT
                    END;
          X:=Y;
          Y:=HOR(Y)
        END
      END;
    LNT:
  END;
  CONT:=+1
END;
BEGIN
  INTEGER ARRAY SAIDA[1:LIMNTER];
  INTEGER I,M;
  FOR I:=1 STEP 1 UNTIL LIMNTER DO
   IF V[I] = 1 THEN BEGIN
                      M:=+1;
                      SAIDA[M]:=I
                    END;
  WRITE(OUTPUT[SKIP 1]);
  IF   M <> 0
  THEN BEGIN
         WRITE(OUTPUT,<///,10("*"),X10,"NAO-TERMINAIS QUE GERAM",
               " SEQUENCIA VAZIA",X10,10("*"),///>);
         FOR I:=1 STEP 1 UNTIL M DO
         IF   SAIDA[I] > NTER
         THEN BEGIN
                NOVONAOTERMINAL(SAIDA[I]);
                WRITE(OUTPUT,<X20,A6,/>,PALAVRA);
              END
         ELSE WRITE(OUTPUT,<X20,"A6,/>,COMP(SAIDA[I]),FOR J:=1
                    STEP 1 UNTIL COMP(SAIDA[I]) DO
                    TABSIMB[INICIO(SAIDA[I])+J-1]);
       END
  ELSE WRITE(OUTPUT,<///,10("*"),X10,"OS NAO-TERMINAIS NAO GER",
             "AM SEQUENCIA VAZIA",X10,10("*")>);
 END;
 FIM:
END;
```

```
%   PROCEDIMENTO QUE IMPRIME A PARTE DE TERMINAIS DA MATRIZ DE          %
%   SIMBOLOS INICIADORES E SEGUIDORES                                   %
%                                                                       %

    PROCEDURE IMPTER(MATRIZ,X,TIPO);
    INTEGER ARRAY MATRIZ[1,1];
    INTEGER X,TIPO;
    COMMENT MATRIZ - MATRIZ CUJOS TERMINAIS SERAO IMPRESSOS
            X      - POSICAO INICIAL DO NAO-TERMINAL NA MATRIZ;
    BEGIN
      INTEGER CT,CNT,F,RF,P,RP,I,J;
      COMMENT CT  - CODIGO DO TERMINAL INICIAL IMPRESSO NA LINHA
              CNT - CODIGO DO NAO-TERMINAL INICIAL IMPRESSO NA COLUNA
              F   - NUMERO DE FOLHAS IMPRESSAS PARA UM MESMO CONJUNTO
                    DE NAO-TERMINAIS IMPRESSOS POR COLUNA
              P   - NUMERO DE PAGINAS IMPRESSAS PARA UM MESMO CONJU
                    DE TERMINAIS IMPRESSOS POR LINHA;
%=================================================================
      PROCEDURE LOOPTER(A,B);
      INTEGER A,B;
      COMMENT A - NUMERO DE TERMINAIS IMPRESSOS POR LINHA
              B - NUMERO DE NAO-TERMINAIS IMPRESSOS POR PAGINA;
      BEGIN
        INTEGER I,J;
        WRITE(OUTPUT[SKIP 1]);
        CASE TIPO OF
        BEGIN
        1:WRITE(OUTPUT,</,X25,"MATRIZ DE SIMBOLOS INICIADORES">);
        2:WRITE(OUTPUT,</,X25,"MATRIZ DE SIMBOLOS SEGUIDORES">);
        END;
        WRITE(OUTPUT,<///,X4,*I4>,A,FOR I:=CT STEP 1 UNTIL CT+A-1 DO
              I+LIMNTER);
        FOR I:=CNT STEP 1 UNTIL CNT+B-1 DO
        BEGIN
          FOR J:=X STEP 1 UNTIL X+A-1 DO Z[J]:=XR(MATRIZ,I,J);
          WRITE(OUTPUT,</,I3,X1,*I4>,I,A,FOR J:=X STEP 1 UNTIL X+A-1 DO
                Z[J])
        END
      END;
%=====================================================================%
      PROCEDURE PAGINA(A);
      INTEGER A;
      COMMENT A - NUMERO DE TERMINAIS IMPRESSOS POR LINHA;
      BEGIN
        INTEGER I,J;
        CNT:=1;
        IF LIMNTER < 40 THEN J:=LIMNTER ELSE J:=40;
        FOR I:=1 STEP 1 UNTIL P DO
        BEGIN
          LOOPTER(A,J);
          CNT:=*+J
        END;
        IF RP ¬= 0 THEN LOOPTER(A,RP)
      END;
%=====================================================================%
      IF    TER+1 <= 40 AND LIMNTER <= 40
      THEN BEGIN
             WRITE(OUTPUT[SKIP 1]);
             CASE TIPO OF
             BEGIN
             1:WRITE(OUTPUT,</,X25,"MATRIZ DE SIMBOLOS INICIADORES">);
             2:WRITE(OUTPUT,</,X25,"MATRIZ DE SIMBOLOS SEGUIDORES">);
             END;
             WRITE(OUTPUT,<///,X3,*I3>,TER+1,FOR I:=1 STEP 1 UNTIL
                   TER+1 DO I+LIMNTER);
             FOR I:=1 STEP 1 UNTIL LIMNTER DO
             BEGIN
               FOR J:=X STEP 1 UNTIL X+TER DO Z[J]:=XR(MATRIZ,I,J);
               WRITE(OUTPUT,</,I2,X1,*I3>,I,TER+1,FOR J:=X STEP 1 UNTIL
                     X+TER DO Z[J])
             END
           END
      ELSE BEGIN
             CT:=1;
             F :=LIMNTER DIV 40;
             RP:=LIMNTER MOD 40;
             F :=(TER+1) DIV 32;
             RF:=(TER+1) MOD 32;
```

```
            FOR I:=1 STEP 1 UNTIL F DO
            BEGIN
              PAGINA(32);
              CT:=*+32;
              X:=*+32
            END;
            IF RF ¬= 0 THEN PAGINA(RF)
          END
    END;
%*****************************************************************%
%*  PROCEDIMENTO QUE IMPRIME A PARTE DE NAO-TERMINAIS DA MATRIZ DE *%
%*  SIMBOLOS INICIADORES E SEGUIDORES                              *%
%*****************************************************************%
    PROCEDURE IMPNTER(MATRIZ,X,TIPO);
    INTEGER ARRAY MATRIZ[1,1];
    INTEGER X,TIPO;
    COMMENT MATRIZ - MATRIZ CUJOS NAO-TERMINAIS SERAO IMPRESSOS
            X      - POSICAO INICIAL DO NAO-TERMINAL NA MATRIZ;
    BEGIN
      INTEGER CIL,CIP,F,RF,P,RP,I,J;
      COMMENT CIL - CODIGO DO NAO-TERMINAL INICIAL IMPRESSO NA LINHA
              CIP - CODIGO DO NAO-TERMINAL INICIAL IMPRESSO NA COLUNA
              F   - NUMERO DE FOLHAS IMPRESSAS PARA UM MESMO CONJUNTO
                    DE NAO-TERMINAIS IMPRESSOS POR COLUNA
              P   - NUMERO DE PAGINAS IMPRESSAS PARA UM MESMO CONJUNTO
                    DE NAO-TERMINAIS IMPRESSOS POR LINHA;
%=================================================================%
      PROCEDURE LOOPNTER(A,B);
      INTEGER A,B;
      COMMENT A - NUMERO DE NAO-TERMINAIS IMPRESSOS POR LINHA
              B - NUMERO DE NAO-TERMINAIS IMPRESSOS POR PAGINA;
      BEGIN
        INTEGER I,J;
        IF CANAL ¬= 1 THEN
        BEGIN
          WRITE(OUTPUT[SKIP 1]);
          CASE TIPO OF
          BEGIN
          1:WRITE(OUTPUT,</,X25,"MATRIZ DE SIMBOLOS INICIADORES">);
          2:WRITE(OUTPUT,</,X25,"MATRIZ DE SIMBOLOS SEGUIDORES">);
          END;
        END;
        WRITE(OUTPUT,<///,X4,*I4>,A,FOR I:=CIL STEP 1 UNTIL CIL+A-1 DO
              I);
        FOR I:=CIP STEP 1 UNTIL CIP+B-1 DO
        BEGIN
          FOR J:=X STEP 1 UNTIL X+A-1 DO Z[J]:=XR(MATRIZ,I,J);
          WRITE(OUTPUT,</,I3,XI,*I4>,I,A,FOR J:=X STEP 1 UNTIL X+A-1 DO
                Z[J])
        END
      END;
%=================================================================%
      PROCEDURE LOOPAGINA(A);
      INTEGER A;
      COMMENT A - NUMERO DE NAO-TERMINAIS IMPRESSOS POR LINHA;
      BEGIN
        INTEGER I;
        CIP:=1;
        FOR I:=1 STEP 1 UNTIL P DO
        BEGIN
          LOOPNTER(A,40);
          CIP:=*+40
        END;
        IF RP ¬= 0 THEN LOOPNTER(A,RP)
      END;
      IF   LIMNTER <= 40
      THEN BEGIN
             IF CANAL ¬= 1 THEN
             BEGIN
               WRITE(OUTPUT[SKIP 1]);
               CASE TIPO OF
               BEGIN
               1:WRITE(OUTPUT,</,X25,"MATRIZ DE SIMBOLOS INICIADORES"
                       >);
               2:WRITE(OUTPUT,</,X25,"MATRIZ DE SIMBOLOS SEGUIDORES">);
               END
             END;
```

```
          WRITE(OUTPUT,<///,X3,'I3>',LIMNTER,'FOR I:=1 STEP 1 UNTIL
                LIMNTER DO I);
          FOR I:=1 STEP 1 UNTIL LIMNTER DO
          BEGIN
            FOR J:=X STEP 1 UNTIL X+LIMNTER-1 DO
              Z[J]:=XR(MATRIZ,I,J);
            WRITE(OUTPUT,<',I2,X1,'I3>',I,LIMNTER,'FOR J:=X STEP 1
                  UNTIL X+LIMNTER-1 DO Z[J])
          END
        END
      ELSE BEGIN
          CIL:=1;
          F :=LIMNTER DIV 32;
          RF:=LIMNTER MOD 32;
          P :=LIMNTER DIV 40;
          RP:=LIMNTER MOD 40;
          FOR CANAL:=1 STEP 1 UNTIL F DO
          BEGIN
            LOOPAGINA(32);
            CIL:=CIL+32;
            X:=X+32
          END;
          IF RF <> 0 THEN LOOPAGINA(RF)
        END
  END;
%                                                                       %
%%  PROCEDIMENTO QUE IMPRIME A MATRIZ DE SIMBOLOS INICIADORES           %%
%                                                                       %
  PROCEDURE OUTSTART;
  BEGIN
    INTEGER I,J;
    INTEGER AUX1, AUX2;
    AUX1:=LIMNTER+1;
    AUX2:=1;
    WRITE(OUTPUT[SKIP 1]);
    WRITE(OUTPUT,<',X25,"MATRIZ DE SIMBOLOS INICIADORES">);
    IF   LIMNTER+TER+1 <= 40
    THEN BEGIN
           WRITE(OUTPUT,<',X12,'I3>',LIMNTER+TER+1,'FOR I:=1 STEP 1
                 UNTIL LIMNTER DO I,'FOR J:=1 STEP 1 UNTIL TER+1 DO
                 J+LIMNTER);
           FOR I:=1 STEP 1 UNTIL LIMNTER DO
           BEGIN
             FOR J:=1 STEP 1 UNTIL 48*N1 DO Z[J]:=S(I,J);
             WRITE(OUTPUT,<',X8,I3,X1,'I3>',I,LIMNTER+TER+1,'FOR J:=1
                   STEP 1 UNTIL LIMNTER+TER+1 DO Z[J])
           END
         END
    ELSE BEGIN
           CANAL:=1;
           IMPNTER (SREAL,AUX2,1);
           IMPTER(SREAL,AUX1,1);
         END
  END;
%                                                                       %
%%  PROCEDIMENTO QUE IMPRIME A MATRIZ DE SIMBOLOS SEGUIDORES           %%
%                                                                       %
  PROCEDURE OUTFOLLOW;
  BEGIN
    INTEGER I,J,K;
    INTEGER AUX1, AUX2, AUX3;
    AUX1:=LIMNTER+1;
    AUX2:=2*LIMNTER+1;
    AUX3:=1;
    WRITE(OUTPUT[SKIP 1]);
    WRITE(OUTPUT,<',X25,"MATRIZ DE SIMBOLOS SEGUIDORES">);
    IF   2*LIMNTER+TER+1 <= 40
    THEN BEGIN
           WRITE(OUTPUT,<',X12,'I3>',2*LIMNTER+TER+1,'FOR I:=1 STEP 1
                 UNTIL LIMNTER DO I,'FOR J:=1 STEP 1 UNTIL LIMNTER DO
                 J,'FOR K:=1 STEP 1 UNTIL TER+1 DO K+LIMNTER);
           FOR I:=1 STEP 1 UNTIL LIMNTER DO
           BEGIN
             FOR J:=1 STEP 1 UNTIL 48*N2 DO Z[J]:=F(I,J);
             WRITE(OUTPUT,<',X8,I3,X1,'I3>',I,2*LIMNTER+TER+1,'FOR J:=1
                   STEP 1 UNTIL 2*LIMNTER+TER+1 DO Z[J])
           END
         END
```

```
  ELSE IF   2*LIMNTER <= 40
        THEN BEGIN
               WRITE(OUTPUT,<//,X12,'I3>,2*LIMNTER,FOR I:=1 STEP 1
                     UNTIL LIMNTER DO I,FOR J:=1 STEP 1 UNTIL
                     LIMNTER DO J);
               FOR I:=1 STEP 1 UNTIL LIMNTER DO
               BEGIN
                 FOR J:=1 STEP 1 UNTIL 2*LIMNTER DO Z[J]:=F[I,J];
                 WRITE(OUTPUT,<//,X8,'I3>,I,2*LIMNTER,FOR J:=1
                       STEP 1 UNTIL 2*LIMNTER DO Z[J])
               END;
               IMPTER(FREAL,AUX2,2);
             END
        ELSE BEGIN
               CANAL:=1;
               IMPNTER(FREAL,AUX3,2);
               CANAL:=2;
               IMPNTER(FREAL,AUX1,2);
               IMPTER(FREAL,AUX2,2);
             END
END;

PROCEDIMENTO QEU IMPRIME OS CONJUNTOS DE SIMBOLOS DIRETORES

PROCEDURE IMPSIMBDIR;
BEGIN
  INTEGER ARRAY A[0:500];
  INTEGER I,J,X;
  WRITE(OUTPUT[SKIP 1]);
  WRITE(OUTPUT,<"[0,"CONJUNTO DE SIMBOLOS DIRETORES - NAO-TERM",
        "INAL  ==> (TERMINAL,NUMERO DA PRODUCAO)",//>);
  FOR I:=1 STEP 1 UNTIL LIMNTER DO
  BEGIN
    INTEGER L,K,M;
    BOOLEAN PARE;
    X:=US[I];
    WHILE X >= 0 DO
    BEGIN
      A[K+1]:=COTER(X);
      A[K+2]:=NPROD(X);
      K:=K+2;
      X:=LINK(X)
    END;
    IF K > 16 THEN L:=16 ELSE BEGIN
                                L:=K;
                                PARE:=TRUE;
                              END;
    WRITE(OUTPUT,<//,I4,"  ==>",'(" ( ",I3," ,",I4," ) ")>,I,L/2,FOR
          J:=1 STEP 1 UNTIL L DO A[J]);
    IF NOT PARE THEN
    BEGIN
    INTEGER C,D;
    C:=(K-16) DIV 16;
    D:=(K-16) MOD 16;
    FOR M:=1 STEP 1 UNTIL C DO
    WRITE(OUTPUT,<//,X8,'(" ( ",I3," ,",I4," ) ")>,FOR J:=16*M+1
          STEP 1 UNTIL 16*M+16 DO A[J]);
    IF D > 0 THEN
    WRITE(OUTPUT,<//,X8,'(" ( ",I3," ,",I4," ) ")>,D/2,
          FOR M:=16*C+17 STEP 1 UNTIL K DO A[M]);
    END;
  END
END;
```

```
%                                                                      %
%   PROCEDIMENTO QUE CALCULA A RELACAO FECHAMENTO TRANSITIVO           %
%                                                                      %
PROCEDURE WARSHALL(A,B,C);
INTEGER ARRAY A[1,1]; INTEGER B,C;
BEGIN
  INTEGER I,J,K;
  DEFINE BIT(I,J) = XR(A,I,J) # ;
  FOR I:=1 STEP 1 UNTIL B DO
  FOR J:=1 STEP 1 UNTIL B DO
    IF   BIT(J,I) = 1
    THEN FOR K:=1 STEP 1 UNTIL C DO
         BIT(J,K):=REAL(BOOLEAN(BIT(J,K)) OR BOOLEAN(BIT(I,K)));
END;

%                                                                      %
%   PROCEDIMENTO QUE CALCULA O CONJUNTO DE SIMBOLOS INICIADORES DE     %
%   CADA NAO-TERMINAL                                                  %
%                                                                      %
PROCEDURE START;
BEGIN
  INTEGER I,X,Y;
  FOR I:=1 STEP 1 UNTIL LIMNTER DO
  BEGIN
    X:=PONT[I];
    WHILE X ¬= 0 DO
    BEGIN
      Y:=PTR(X);
      IF Y ¬= 0 THEN
      BEGIN
        IF   CODIGO(Y) > MAXNTER
        THEN S[I,CODIGO(Y)-MAXNTER+LIMNTER):=1
        ELSE BEGIN
               LABEL L1;
               S[I,CODIGO(Y)]:=1;
               WHILE CODIGO(Y) <= MAXNTER AND RLINK(Y) ¬= 0 DO
                IF   V[CODIGO(Y)] ¬= 1 THEN GO L1
                ELSE BEGIN
                       Y:=RLINK(Y);
                       S[I,CODIGO(Y)]:=1
                     END;
               L1:
             END
      END;
      X:=LLINK(X)
    END
  END;
  WARSHALL(SREAL,LIMNTER,LIMNTER+TER+1);
END;

%                                                                      %
%   PROCEDIMENTO QUE CALCULA A FUNCAO LAST PARA UM NAO-TERMINAL        %
%                                                                      %
PROCEDURE LAST(A);
INTEGER A;
BEGIN
  INTEGER X,Y;
  X:=PONT[A];
  WHILE X ¬= 0 DO
  BEGIN
    IF PTR(X) ¬= 0 THEN
    BEGIN
      INTEGER ARRAY VETOR[1:PRODNTER+2]; INTEGER I;
      Y:=PTR(X);
      WHILE Y ¬= 0 DO BEGIN
                        I:=I+1;
                        VETOR[I]:=Y;
                        Y:=RLINK(Y)
                      END;
      IF CODIGO(VETOR[I]) <= MAXNTER THEN
      BEGIN
        F[CODIGO(VETOR[I]),A+LIMNTER):=1;
        WHILE I > 1 AND CODIGO(VETOR[I]) <= MAXNTER DO
          IF   V[CODIGO(VETOR[I])] ¬= 1 THEN I:=I
          ELSE BEGIN
                 I:=I-1;
                 IF   CODIGO(VETOR[I]) <= MAXNTER
                 THEN F[CODIGO(VETOR[I]),A+LIMNTER):=1
               END;
      END
```

```
      END;
      X:=LLINK(X)
    END
  END;

%   PROCEDIMENTO QUE CALCULA O CONJUNTO DE SIMBOLOS SEGUIDORES DE    %
%   CADA NAO-TERMINAL                                                %

  PROCEDURE FOLLOW;
  BEGIN
    INTEGER I,J,K,X,Y,A;
    INTEGER ARRAY MATRIZ[1:MAXNTER,1:N1];
    DEFINE M(I,J) = XR(MATRIZ,I,J) #;
    F(1,2*LIMNTER+TER+1):=1;
    FOR I:=1 STEP 1 UNTIL LIMNTER DO
    BEGIN
      X:=PONT[I];
      WHILE X ¬= 0 DO
      BEGIN
        Y:=PTR(X);
        WHILE Y ¬= 0 DO
        BEGIN
          IF CODIGO(Y) <= MAXNTER AND RLINK(Y) ¬= 0 THEN
          BEGIN
            LABEL L1;
            A:=RLINK(Y);
            IF   CODIGO(A) > MAXNTER
            THEN F(CODIGO(Y),CODIGO(A)-MAXNTER+2*LIMNTER):=1
            ELSE F(CODIGO(Y),CODIGO(A)):=1;
            LAST(CODIGO(Y));
            WHILE RLINK(A) ¬= 0 AND CODIGO(A) <= MAXNTER DO
             IF   V[CODIGO(A)] ¬= 1
             THEN GO L1
             ELSE BEGIN
                    A:=RLINK(A);
                    IF   CODIGO(A) <= MAXNTER
                    THEN F(CODIGO(Y),CODIGO(A)):=1
                    ELSE F(CODIGO(Y),CODIGO(A)-MAXNTER+2*LIMNTER):=1
                  END;
            L1:
          END
          ELSE IF   RLINK(Y) = 0
               THEN IF   CODIGO(Y) <= MAXNTER AND CODIGO(Y) ¬= I
                    THEN F(CODIGO(Y),I+LIMNTER):=1;
          Y:=RLINK(Y)
        END;
        X:=LLINK(X)
      END
    END;
    FOR I:=1 STEP 1 UNTIL LIMNTER DO FOR J:=1 STEP 1 UNTIL LIMNTER DO
     IF   F(J,I) = 1
     THEN FOR K:=2*LIMNTER+1 STEP 1 UNTIL 2*LIMNTER+TER+1 DO
            F(J,K):=REAL(BOOLEAN(F(J,K)) OR BOOLEAN(S(I,K-LIMNTER)));
    FOR I:=1 STEP 1 UNTIL LIMNTER DO
     FOR J:=1 STEP 1 UNTIL LIMNTER+TER+1 DO M(I,J):=F(I,J+LIMNTER);
    WARSHALL(MATRIZ,LIMNTER,LIMNTER+TER+1);
    FOR I:=1 STEP 1 UNTIL LIMNTER DO
     FOR J:=1 STEP 1 UNTIL LIMNTER+TER+1 DO F(I,J+LIMNTER):=M(I,J);
  END;
```

```
%*****  PROCEDIMENTO QUE ELIMINA RECURSAO A ESQUERDA DA GRAMATICA  *****%
  PROCEDURE ELIMINERECURSAO(N,W);
  INTEGER ARRAY W[1];
  INTEGER N;
  COMMENT N - NUMERO DE NAO-TERMINAIS RECURSIVOS
          W - VETOR DO GRUPO RECURSIVO;
  BEGIN
    LABEL FINAL;
    INTEGER I,J,K,Y,A,B,C,IND;
    DEFINE Z(I,J) = [0]M[ER + I*N + J - N #,
           X(I)   = W[I] #;
%=================================================================%
    PROCEDURE PEGUENO(X);
    INTEGER X;
    BEGIN
      IND:=0;
      IF   PTR(DISP) =: 0
      THEN BEGIN
             X:=DISP;
             DISP:=PTR(X)
           END
      ELSE BEGIN
             IND:=1;
             WRITE(OUTPUT,<///,X10,"ACABOU NO' DISPONIVEL">)
           END
    END;
%=================================================================%
    PROCEDURE COPIEPROD(X,Y,Z);
    INTEGER X,Y,Z;
    COMMENT X - INICIO DA PRODUCAO
            Y - NO' INICIAL DA COPIA
            Z - NO' FINAL DA COPIA;
    BEGIN
      INTEGER P1,P2;
      LABEL FIM;
      PEGUENO(P1);
      IF IND = 1 THEN GO FIM;
      Y:=P1;
      IF   RLINK(X) = 0
      THEN BEGIN
             CODIGO(P1):=CODIGO(X);
             Z:=P1
           END
      ELSE BEGIN
             WHILE RLINK(X) =: 0 DO
             BEGIN
               PEGUENO(P2);
               IF IND = 1 THEN GO FIM;
               CODIGO(P1):=CODIGO(X);
               RLINK(P1):=P2;
               P1:=P2;
               X:=RLINK(X)
             END;
             CODIGO(P2):=CODIGO(X);
             Z:=P2;
           END;
      FIM:
    END;
%=================================================================%
    PROCEDURE CRIEPRODUCAO(A,B,C);
    INTEGER A,B,C;
    COMMENT A - CODIGO DE NAO-TERMINAL
            B - CODIGO DE NAO-TERMINAL
            C - INICIO DA PRODUCAO;
    BEGIN
      INTEGER P1,P2,E,F;
      LABEL FIM;
      PEGUENO(P1);
      IF IND = 1 THEN GO FIM;
      IF   PONT[A] = 0
      THEN BEGIN
             PONT[A]:=P1;
             PINT(A):=1
           END
      ELSE LLINK(INP(A)):=P1;
```

```
      INP(A):=PT1;
      LLINK(PT1):=0;
      IF    C ¬= 0
      THEN BEGIN
             COPIEPROD(C,E,F);
             RLINK(PT1):=E
           END
      ELSE F:=PT1;
      IF    B ¬= 0
      THEN BEGIN
             PEGUEND(PT2);
             IF IND = 1 THEN GO FIM;
             RLINK(F):=PT2;
             CODIGO(PT2):=B;
             RLINK(PT2):=0
           END
      ELSE RLINK(PT1):=0;
      FIM:
    END;
%=============================================================%
    FOR I:=1 STEP 1 UNTIL N DO
    BEGIN
      J:=PONTEIRO[I];
      K:=1;
      WHILE LLINK(J) ¬= 0 DO
      BEGIN
        J:=LLINK(J);
        K:=K+1
      END;
      PINT[W[I]]:=K;
      INP[W[I]]:=J
    END;
    COMMENT PINT - NUMERO DE PRODUCOES DO NAO-TERMINAL ANTES DA
                   ELIMINACAO DA RECURSAO
            INP  - PONTEIRO PARA A ULTIMA PRODUCAO DO NAO-TERMINAL;
    FOR I:=1 STEP 1 UNTIL N DO
    BEGIN
      LABEL FIM;
      A:=PONTEIRO[I];
      FOR C:=1 STEP 1 UNTIL PINT[W[I]] DO
      IF    PTR(A) ¬= 0
      THEN BEGIN
             Y:=PTR(A);
             IF    CODIGO(Y) <= MAXNTER
             THEN FOR J:=1 STEP 1 UNTIL N DO
                  IF CODIGO(Y) = W[J] THEN
                  BEGIN
                    Y:=RLINK(Y);
                    FOR K:=1 STEP 1 UNTIL N DO
                     CRIEPRODUCAO(Z(J,K),Z(I,K),Y);
                    GO FIM
                  END;
             FOR K:=1 STEP 1 UNTIL N DO
              CRIEPRODUCAO(X(K),Z(I,K),Y);
             FIM:
             IF IND = 1 THEN GO FINAL;
             B:=A;
             A:=LLINK(A);
             PONTEIRO[W[I]]:=LLINK(B);
           END
      ELSE BEGIN
             FOR K:=1 STEP 1 UNTIL N DO
              CRIEPRODUCAO(X(K),Z(I,K),0);
             IF IND = 1 THEN GO FINAL;
             B:=A;
             A:=LLINK(A);
             PONTEIRO[W[I]]:=LLINK(B);
           END;
      CRIEPRODUCAO(Z(I,I),0,0)
    END;
    TOTINTER:=N+N**2;
    FINAL:
  END;
```

```
{*****
  PROCEDIMENTO QUE DETERMINA OS GRUPOS DE NAO-TERMINAIS RECURSIVOS
  ENTRE SI
*****}

PROCEDURE GRUPOSRECURSIVOS;
BEGIN
  INTEGER ARRAY GR,IND[1:LIMNTER];
  INTEGER ARRAY W[1:LIMNTER];
  INTEGER I,J,K,X;
  FOR I:=1 STEP 1 UNTIL LIMNTER DO GR[I]:=I;
  FOR I:=1 STEP 1 UNTIL LIMNTER DO
   IF   S[I,I] = 1
   THEN BEGIN
          IND[I]:=1;
          X:=X+1
        END;
  IF   X <> 0
  THEN BEGIN
         WRITE(OUTPUT,<///,10("*"),X10,"GRUPO DE NAO-TERMINAIS ",
               "RECURSIVOS",X10,10("*"),//>);
       END;
  WHILE X > K DO
  BEGIN
    FOR I:=1 STEP 1 UNTIL LIMNTER DO
     IF   IND[I] = 1
     THEN FOR J:=GR[I] STEP 1 UNTIL LIMNTER DO
            IF IND[J] = 1 THEN
            BEGIN
              IF   S(GR[I],J) <> 1
              THEN IND[J]:=2
              ELSE IF   S(J,GR[I]) <> 1
                   THEN IND[J]:=2
            END;
    J:=0;
    FOR I:=1 STEP 1 UNTIL LIMNTER DO
     IF IND[I] <> 0 THEN
     BEGIN
       IF   IND[I] = 1
       THEN BEGIN
              K:=K+1;
              J:=J+1;
              W[J]:=GR[I];
              IND[I]:=0
            END
       ELSE IF   IND[I] = 2
            THEN IND[I]:=1
     END;
    IF   J <> 0
    THEN BEGIN
           ALPHA ARRAY Z[1:13*NTER];
           INTEGER A,B;
           FOR I:=1 STEP 1 UNTIL J DO
           BEGIN
             IF   W[I] > NTER
             THEN BEGIN
                    NOVONAOTERMINAL(W[I]);
                    B:=B+1;
                    Z[B]:=PALAVRA;
                  END
             ELSE FOR A:=1 STEP 1 UNTIL COMP(W[I]) DO
                  BEGIN
                    B:=B+1;
                    Z[B]:=TABSIMB[INICIO(W[I])+A-1]
                  END;
             B:=B+1;
             Z[B]:="       ";
           END;
           WRITE(OUTPUT,<X20,18A6,/>,FOR A:=1 STEP 1 UNTIL B DO
                 Z[A]);
         END;
    ELIMINERECURSAO(J,W);
  END;
END;
```

```
{*****
    PROCEDIMENTO QUE VERIFICA SE A GRAMATICA APRESENTA NAO-TERMINAIS
    RECURSIVOS A ESQUERDA                                              *****}

    PROCEDURE DETETARECURSAO;
    BEGIN
      INTEGER ARRAY A,NTA[1:MAXNTER];
      INTEGER I,J;  LABEL  FOR;
      DEFINE A1(I) = A[I].[47:24] #;
             A2(I) = A[I].[23:24] #;
      PROCEDURE NTERALCANSAVEIS;
      COMMENT - PROCEDIMENTO QUE DETERMINA OS NAO-TERMINAIS ALCANSAVEIS
                APOS A ELIMINACAO DA RECURSAO;
      BEGIN
        INTEGER K,X,Y,P,P1,P2;  LABEL L1;
        K:=P:=1;
        NTA[1]:=1;
        A1[1]:=1;
        WHILE NTA[P] ¬= 0 DO
        BEGIN
          X:=PONT[NTA[P]];
          WHILE X ¬= 0 DO
          BEGIN
            Y:=PTR(X);
            WHILE Y ¬= 0 DO
            BEGIN
              IF   CODIGO(Y) <= MAXNTER
              THEN IF   A1(CODIGO(Y)) = 0
                   THEN BEGIN
                          K:=K+1;
                          NTA[K]:=CODIGO(Y);
                          A1(CODIGO(Y)):=CODIGO(Y);
                        END;
              Y:=RLINK(Y);
            END;
            X:=LLINK(X)
          END;
          P:=P+1
        END;
        TAMNTER:=K;
        IF TAMNTER = K THEN GO L1;
        P1:=1;
        P2:=TAMNTER;
        WHILE P1 < P2 DO
        BEGIN
          WHILE A1(P1) ¬= 0 AND P1 <= K DO
          BEGIN
            A2(P1):=PONT[P1];
            P1:=P1+1
          END;
          WHILE A1(P2) = 0 AND P2 > K DO P2:=P2-1;
          A1(P1):=P2;
          A1(P2):=P1;
          A2(P1):=PONT[P2];
          P1:=P1+1;
          P2:=P2-1;
        END;
        FOR P:=1 STEP 1 UNTIL K DO
        BEGIN
          X:=PONT[NTA[P]];
          WHILE X ¬= 0 DO
          BEGIN
            Y:=PTR(X);
            WHILE Y ¬= 0 DO
            BEGIN
              IF   CODIGO(Y) > K AND CODIGO(Y) <= MAXNTER
              THEN CODIGO(Y):=A1(CODIGO(Y));
              Y:=RLINK(Y)
            END;
            X:=LLINK(X)
          END
        END;
        FOR P:=1 STEP 1 UNTIL K DO
        BEGIN
          PONT[P]:=A2(P);
          VNT[P]:=A2(P);
        END;
```

```
    L1:
    FOR P:=1 STEP 1 UNTIL K DO VNT[P]:=NTA[P];
  END;
  RECURSAO:=FALSE;
  FOR I:=1 STEP 1 UNTIL LIMNTER DO
    IF   S[I,I] = 1
    THEN BEGIN
           J:=1;
           GO FORA
         END;
  FORA:
  IF   J ¬= 0
  THEN BEGIN
         RECURSAO:=TRUE;
         GRUPOSRECURSIVOS;
         NTERALCANSAVEIS;
       END
  ELSE BEGIN
       WRITE(OUTPUT,<///,X10,"OS NAO-TERMINAIS NAO APRESENTAM RECU",
             "RSAO A ESQUERDA">);
       END;
END;

PROCEDIMENTO QUE DETERMINA OS SIMBOLOS DIRETORES DA GRAMATICA

PROCEDURE SIMBDIR;
BEGIN
  INTEGER ARRAY CONF[0:1000];
  INTEGER I,J,K,NP,IND,AVAIL,X,Y;
  LABEL FINAL;
  PROCEDURE CONFLITO(A);
  INTEGER A;
  COMMENT - PROCEDIMENTO QUE VERIFICA A EXISTENCIA DE CONFLITOS NA
            TABELA DE SIMBOLOS DIRETORES;
  BEGIN
    INTEGER I,J,M,V;
    LABEL FIM;
    FOR J:=1 STEP 1 UNTIL TER+1 DO
      DIR(1,J):= REAL(BOOLEAN(DIR(3,J)) AND BOOLEAN(DIR(2,J)));
    FOR I:=1 STEP 1 UNTIL TER DIV 48 + 1 DO
      IF   LADO[DIR(1,I)] ¬= 0
      THEN FOR J:=48*(I-1)+1 STEP 1 UNTIL 48*I DO
             IF   DIR(1,J) = 1
             THEN BEGIN
                    CONF[K]:=A;
                    CONF[K+1]:=J+LIMNTER;
                    CONF[K+2]:=NP;
                    Y:=DS[A];
                    WHILE M ¬= 0 DO
                    BEGIN
                      IF   J+LIMNTER = CODTER(M)
                      THEN CONF[K+3]:=NPROD(M);
                      M:=LINK(M)
                    END;
                    K:=K+4;
                  END;
    FOR J:=TER+1 STEP -1 UNTIL 1 DO
      IF   DIR(2,J) = 1
      THEN BEGIN
             IND:=0;
             IF   LINK(AVAIL) ¬= 0
             THEN BEGIN
                    V:=AVAIL;
                    AVAIL:=LINK(V)
                  END
             ELSE BEGIN
                    IND:=1;
                    WRITE(OUTPUT,<///,X10,"ACABOU NODO DISPONIVEL">);
                    GO FIM
                  END;
             CODTER(V):=J+LIMNTER;
             NPROD(V):=NP;
             LINK(V):=DS[A];
             DS[A]:=V;
           END;
    FOR J:=1 STEP 1 UNTIL TER+1 DO
```

```
      DIR(3,J):= REAL(BOOLEAN(DIR(3,J)) OR BOOLEAN(DIR(2,J)));
    FIN:
  END;
  AVAIL:=1;
  FOR I:=1 STEP 1 UNTIL 10*MAXNTER-1 DO LINK(I):=I+1;
  LINK(10*MAXNTER):=0;
  FOR I:=1 STEP 1 UNTIL LIMNTER DO
  BEGIN
    X:=PONT(I);
    WHILE X ¬= 0 DO
    BEGIN
      NP:=NP+1;
      IF   PTR(X) = 0
      THEN FOR J:=2*LIMNTER+1 STEP 1 UNTIL 2*LIMNTER+TER+1 DO
           DIR(2,J-2*LIMNTER):=F(I,J)
      ELSE BEGIN
             Y:=PTR(X);
             IF   CODIGO(Y) > MAXNTER
             THEN DIR(2,CODIGO(Y)-MAXNTER):=1
             ELSE BEGIN
                    LABEL SAI;
                    FOR J:=LIMNTER+1 STEP 1 UNTIL LIMNTER+TER+1 DO
                     DIR(2,J-LIMNTER):=S(CODIGO(Y),J);
                    WHILE RLINK(Y) ¬= 0 AND CODIGO(Y) <= MAXNTER DO
                    IF   V(CODIGO(Y)) ¬= I
                    THEN GO SAI
                    ELSE BEGIN
                           Y:=RLINK(Y);
                           FOR J:=LIMNTER+1 STEP 1 UNTIL
                            LIMNTER+TER+1 DO
                            DIR(2,J-LIMNTER):=
                            REAL(BOOLEAN(DIR(2,J-LIMNTER)) OR
                                 BOOLEAN(S(CODIGO(Y),J)))
                         END;
                    SAI:
                  END
           END;
      CONFLITO(I);
      IF IND = 1 THEN GO FINAL;
      FOR J:=1 STEP 1 UNTIL N3 DO LADODIR(2,J):=0;
      X:=LLINK(X)
    END;
    FOR J:=1 STEP 1 UNTIL N3 DO LADODIR(3,J):=0;
  END;
  IMPSIMBDIR;
  WRITE(OUTPUT[SKIP 1]);
  IF   CONF(0) ¬= 0
  THEN BEGIN
         WRITE(OUTPUT,<//,X10,"A GRAMATICA NAO E' LL(1) EM DECO",
                      "RRENCIA DOS CONFLITOS ABAIXO",//>);
         WRITE(OUTPUT,<X10,"(NAO-TERMINAL,TERMINAL)==>PRODUCOES">);
         FOR I:=0 STEP 4 WHILE CONF(I) ¬= 0 DO
         WRITE(OUTPUT,</,X10,"(  ",I3,"  ,",I3,"  ) ==>",I5,"  ,",I5>,
               CONF(I),CONF(I+1),CONF(I+2),CONF(I+3));
       END
  ELSE WRITE(OUTPUT,<//,X25,80("*"),/,X25,"*",T105,"*",/,X25,"*",
             X23,"A  G R A M A T I C A  E'  L L ( 1 )",T105,"*",/,
             X25,"*",T105,"*",/,X25,80("*")>);
  FINAL:
END;
```

```
%    PROCEDIMENTO QUE DA UMA REPRESENTACAO INTERNA A GRAMATICA LIDA    %
%
     PROCEDURE CODIFICADOR;
     BEGIN
       INTEGER I,J,K,L,R,PROX,CLASSE,TIPO,NPAL,COD,PTS,A1,A2,X1,X2,IFN;
       INTEGER CONT,M;
       ALPHA CHAR;
       LABEL TERMINE;
       BOOLEAN FIMSCANER,BLACK,BIDLE;
       ALPHA ARRAY CARTAO[1:80],
                   DADOS[1:13];
       INTEGER ARRAY TABELACO:255],
                     NOINDC1:NTER],
                     PNTC1:MAXNTER];
%=============================================================================%
       PROCEDURE GETCHAR;
       COMMENT - PROCEDIMENTO QUE ACHA O PROXIMO CARACTER E DETERMINA
                 SUA CLASSE;
       BEGIN
         COMMENT -                          CLASSE
                    ESTRELA (*)               1
                    DIGITO                    2
                    LETRA                     3
                    BRANCO                    4
                    IGUAL   (=)               5
                    BARRA   (/)               6
                    PONTO E VIRGULA           7
                    OPERADOR                  8
                    JOGO DA VELHA (#)         9
                    PLICA   (')              10
                    SIMBOLO INVALIDO         11
         PROX:=PROX+1;
         IF PROX <= 80 THEN
         BEGIN
           CHAR:=CARTAO[PROX];
           CLASSE:=TABELACCHAR.[7:8]];
           IF CLASSE = 0 THEN CLASSE:=11
         END
       END;
%=============================================================================%
       PROCEDURE GETSPACE;
       COMMENT - PROCEDIMENTO QUE ACHA UM CARACTER DIFERENTE DE BRANCO,
                 COMENTARIO E MUDALINHA;
       BEGIN
         INTEGER X;
         LABEL FIM;
         X:=PROX;
         WHILE CHAR = 64 AND PROX <= 80 DO GETCHAR;
         IF X = PROX
         THEN BLACK:=TRUE
         ELSE BEGIN
                IF PROX <= 80 THEN
                BEGIN
                  IF   CLASSE = 9
                  THEN PROX:=81
                  ELSE IF   CLASSE >= 3 AND CLASSE <= 10 AND CLASSE <= 7
                            AND CLASSE >= 6
                       THEN BEGIN
                              BLACK:=TRUE;
                              GO FIM
                            END;
                END;
                BLACK:=FALSE;
                TIPO:=10
              END;
         FIM:
       END;
%=============================================================================%
       PROCEDURE ERRO1;
       BEGIN
       COMMENT - PROCEDIMENTO QUE INDICA O TIPO DE ERRO LEXICO;
         WRITE(OUTPUT,<"X10,"SIMBOLO INVALIDO">);
         WHILE CLASSE >= 4 DO GETCHAR
       END;
```

```
PROCEDURE SCAN;
BEGIN
  IF PROX < 80 THEN
  BEGIN
    GETSPACE;
    IF BLACK THEN
    BEGIN
      FOR I:=1 STEP 1 UNTIL 13 DO DADO[I]:="
      L:=0; I:=1; K:=47;
      CASE CLASSE OF
      BEGIN
        5:BEGIN
            TIPO:=1;
            GETCHAR
          END;
        6:BEGIN
            TIPO:=2;
            GETCHAR
          END;
        7:BEGIN
            TIPO:=3;
            GETCHAR
          END;
        3:BEGIN
            WHILE CLASSE <= 3 OR CLASSE = 8 DO
            IF   L < 6
            THEN BEGIN
                   L:=L+1;
                   DADO[I].[K:8]:=CHAR;
                   K:=K-8;
                   GETCHAR
                 END
            ELSE BEGIN
                   L:=0; K:=47;
                   I:=I+1
                 END;
            IF   CLASSE = 4 OR CLASSE = 5
            THEN BEGIN
                   TIPO:=3;
                   NPAL:=I
                 END
            ELSE ERRO1;
          END;
        10:BEGIN
            GETCHAR;
            WHILE CLASSE < 10 DO
            IF   L < 6
            THEN BEGIN
                   L:=L+1;
                   DADO[I].[K:8]:=CHAR;
                   K:=K-8;
                   GETCHAR
                 END
            ELSE BEGIN
                   L:=0; K:=47;
                   I:=I+1
                 END;
            IF   CLASSE = 10
            THEN BEGIN
                   TIPO:=2;
                   GETCHAR;
                   NPAL:=I
                 END
            ELSE ERRO1;
          END;
        11:ERRO1
      END
    END
  END
END;
```

```
  PROCEDURE SCANNER;
  BEGIN
    COMMENT -                                        TIPO
              IGUAL (=)                               1
              BARRA (|)                               2
              PONTO E VIRGULA                         3
              NAO-TERMINAL (IDENTIFICADOR)            8
              TERMINAL (CADEIA)                       9
              ESPACO                                 10
    LABEL L10,FIM;
    IF   PROX >= 80
    THEN BEGIN
           CONT:=CONT+1;
           READ(INPUT,<80A1>,CARTAO[CONT])[L10];
           WRITE(OUTPUT,</,[5,X3,80A1>,CONT,CARTAO[CONT]);
           PROX:=CLASSE:=TIPO:=0;
           CHAR:=" ";
         END;
    SCAN;
    GO FIM;
    L10:FIMSCANER:=TRUE;
    FIM:
  END;
%=============================================================%
  PROCEDURE ERRO(K);
  INTEGER K;
  COMMENT - PROCEDIMENTO QUE INDICA O TIPO DE ERRO SINTATICO;
  BEGIN
    CASE K OF
    BEGIN
      1:WRITE(OUTPUT,</,X10,"ESPERADO UM ESPACO ANTES DA REGRA">);
      2:WRITE(OUTPUT,</,X10,"ESPERADO UM ESPACO APOS A REGRA">);
      3:WRITE(OUTPUT,</,X10,"PARTE ESQUERDA INVALIDA">);
      4:WRITE(OUTPUT,</,X10,"ESPERADO UM '=' APOS A PARTE ESQUERDA"
          >);
      5:WRITE(OUTPUT,</,X10,"ESPERADO UM ';' APOS AS ALTERNATIVAS"
          >);
      6:WRITE(OUTPUT,</,X10,"PARTE DIREITA INVALIDA">);
    END;
    WRITE(OUTPUT,</,X50,"***** EXECUCAO SUSPENSA">);
    FLAG:=TRUE
  END;
%=============================================================%
  PROCEDURE PEGUENO(A);
  INTEGER A;
  COMMENT - PROCEDIMENTO QUE RETIRA UM NO' DA LISTA DE NOS
            DISPONIVEIS;
  BEGIN
    IF   PTR(DISP) <> 0
    THEN BEGIN
           A:=DISP;
           DISP:=PTR(A)
         END
    ELSE A:=0
  END;
%=============================================================%
  PROCEDURE CODIFIQUE(X,Y,Z);
  INTEGER X,Y,Z;
  COMMENT - PROCEDIMENTO QUE DA UM CODIGO AO NAO-TERMINAL OU
            TERMINAL IDENTIFICADO;
  BEGIN
    LABEL FIM;
    TEM:=0;
    PEGUENO(Y);
    IF Y = 0 THEN BEGIN
                    TEM:=1;
                    GO FIM
                  END;
    PTR(X):=Y;
    CODIGO(Y):=Z;
    RLINK(Y):=0;
    FIM:
  END;
```

```
    PROCEDURE TABELADESIMBOLOS(X,Y,Z);
    INTEGER X,Y,Z;
    COMMENT - PROCEDIMENTO QUE COLOCA O TERMINAL OU NAO-TERMINAL
              IDENTIFICADO EM UMA TABELA;
    BEGIN
      INTEGER I,J,L;  LABEL FORA;
      BOOLT:=FALSE;
      IF Y >= X THEN
       FOR I:=X STEP 1 UNTIL Y DO
       IF   COMP(I) = NPAL
       THEN BEGIN
              FOR J:=1 STEP 1 UNTIL NPAL DO
              IF   DADO(J) <> TABSIMB(J+INICIO(I)-1)
              THEN GO FORA;
              BOOLT:=TRUE;
              Z:=I;
              FORA:
            END;
      IF   NOT BOOLT
      THEN BEGIN
             Z:=Y+1;
             INICIO(Y+1):=PTS+1;
             COMP(Y+1):=NPAL;
             FOR J:=1 STEP 1 UNTIL NPAL DO TABSIMB(PTS+J):=DADO(J);
             PTS:=PTS+NPAL;
           END;
    END;
%=====================================================================%
    PROCEDURE PARTEDIREITA;
    BEGIN
      INTEGER Y;  LABEL FIM;
      Y:=TIPO;
      SCANER;
      X1:=A1;
      WHILE Y = 10 AND (TIPO = 8 OR TIPO = 9) DO
      BEGIN
        CASE TIPO OF
        BEGIN
          8:BEGIN
              TABELADESIMBOLOS(1,NTER,COD);
              IF NOT BOOLT THEN NTER:=NTER+1
            END;
          9:BEGIN
              TABELADESIMBOLOS(MAXNTER+1,MAXNTER+TER,COD);
              IF NOT BOOLT THEN TER:=TER+1
            END
        END;
        CONFIGURE(X1,X2,COD);
        X1:=X2;
        IF IFN = 1 THEN GO FIM;
        SCANER;
        Y:=TIPO;
        SCANER;
      END;
      IF Y <> 10 THEN ERRO(6) ELSE PTR(X1):=0;
      IF TIPO = 10 THEN SCANER;
      FIM:
    END;
%=====================================================================%
    PROCEDURE ALTERNATIVAS;
    BEGIN
      LABEL FIM;
      PARTEDIREITA;
      IF FLAG OR IFN = 1 THEN GO FIM;
      WHILE TIPO = 2 DO
      BEGIN
        PEGUENO(A2);
        IF A2 = 0 THEN GO FIM;
        LLINK(A1):=A2;
        LLINK(A2):=0;
        NCARD(A2):=CONT;
        A1:=A2;
        SCANER;
        PARTEDIREITA;
        IF FLAG THEN GO FIM;
      END;
      FIM:
    END;
```

```
    PROCEDURE REGRA;
    BEGIN
      LABEL FIM;
      BOOLEAN BIS;
      IF   TIPO = R
      THEN BEGIN
             TABELADESIMBOLOS(1,NTER,COD);
             IF NOT BOOLF THEN NTER:=NTER+1;
             SCANER;
             IF   TIPO = 1
             THEN BEGIN
                    REGUERDO(A1);
                    IF A1 = 0 THEN GO FIM;
                    IF   BOOLF
                    THEN BEGIN
                           LABEL L1; INTEGER X,Y;
                           FOR X:=1 STEP 1 UNTIL R DO
                           IF   VNT(X) =.COD
                           THEN BEGIN
                                  Y:=PNT[X];
                                  WHILE LLINK(Y) <> 0 DO
                                    Y:=LLINK(Y);
                                  LLINK(Y):=A1;
                                  BIS:=TRUE;
                                  GO L1;
                                END;
                           L1:
                         END;
                    IF   NOT BIS
                    THEN BEGIN
                           R:=R+1;
                           VNT[R]:=COD;
                           PNT[R]:=A1;
                           LLINK(A1):=0;
                         END;
                    NEXPO(A1):=CONT;
                    SCANER;
                    ALTERNATIVAS;
                    IF FLAG THEN GO FIM;
                    IF   TIPO <> 3
                  $ THEN ERRO(5)
                    ELSE SCANER
                  END
             ELSE ERRO(4)
           END
      ELSE ERRO(3);
      FIM:
    END;
%=============================================================%
    PROCEDURE PROG;
    BEGIN
      LABEL FIM;
      SCANER;
      IF   TIPO = 10
      THEN WHILE NOT FIMSCANER DO
           BEGIN
             SCANER;
             REGRA;
             IF FLAG THEN GO FIM;
             IF TIPO <> 10 THEN ERRO(2)
                           ELSE SCANER;
           END
      ELSE ERRO(1);
      FIM:
    END;
%=============================================================%
    FILL TABELA[*] WITH 64(0),4,13(0),4(8),6,12(0),1,8,7,3(8),9(0),8,
                        2(0),8,11(0),8,9,4,10,5,56(0),   9(3),7(0),
                        9(3),3(0),8(3),6(0),13(2),6(0);
    DISP:=1;
    FOR I:=DISP STEP 1 UNTIL POOL-1 DO RLINK(I):=I+1;
    RLINK(POOL):=0;
    PROX:=A1;
```

```
  WRITE(OUTPUT,<X25,80("*"),/,X25,"*",T105,"*",/,X25,"*",X25,
        "GRAMATICA LIDA",T105,"*",/,X25,"*",T105,"*",/,X25,80("*"),
        ///," OBS.: A NUMERACAO AO LADO DAS PRODUCOES CORRESPON",
        "DE AO NUMERO DO CARTAO LIDO",//>);
  GLC;
  IF FLAG THEN GO TERMINE;
  FOR I:=1 STEP 1 UNTIL NTER DO
  IF   VN[I] = 0
  THEN BEGIN
         M:=M+1;
         NTIND[M]:=I
       END
  ELSE BEGIN
         LABEL F1;
         IF   VN[I] = I
         THEN PONT[I]:=PNT[I]
         ELSE FOR J:=1 STEP 1 UNTIL NTER DO
              IF VN[J] = I THEN BEGIN
                                  PONT[I]:=PNT[J];
                                  GO F1;
                                END;
         F1:
       END;
  IF   M <> 0
  THEN BEGIN
         WRITE(OUTPUT,<//,X10,"NAO-TERMINAIS QUE NAO POSSUEM LADO",
               " DIREITO",//>);
         FOR I:=1 STEP 1 UNTIL M DO WRITE(OUTPUT,<X10,A6,//>,
          COMP(NTIND[I]),FOR J:=1 STEP 1 UNTIL COMP(NTIND[I]) DO
          TABSIMB[INICIO(NTIND[I])+J-1]);
         FLAG:=TRUE;
       END;
  LIMNTER:=NTER;
  TERMINE:
END;
```

```
BEGIN
  LABEL PARE;
  DEFINE IND(1) = INDICADOR.[1:8] #;
  IF IND(23) = 123 THEN
  BEGIN
    CASE IND(15) OF
    BEGIN
    198:BEGIN
          FAT:=TRUE;
          IF IND(7) = 217 THEN REC:=TRUE
        END;
    217:REC:=TRUE;
    211:LLI:=TRUE;
    237:TR :=TRUE;
    END;
    READ(INPUT,<SLI>,RI,GC,GF,GR,TAB);
    MODIFICADOR;
    IF FLAG THEN GO PARE;
    IF   RI THEN GRAMATICACODIFICADA
    ELSE WRITE(OUTPUT,<///,1X,"*",X10,"IMPRESSAO DA REPRESENTACAO",
                 " INTERNA DA GRAMATICA SUSPENSA PELO USUARIO",X10,
                 10("*"),>);
    IF   GC THEN GRAMATICAFINAL(1)
    ELSE WRITE(OUTPUT,<///,1)("*"),X10,"IMPRESSAO DA GRAMATICA COD",
                 "IFICADA SUSPENSA PELO USUARIO",X10,10("*"),>);
    IF  FAT OR TR  THEN DETETAFATORRESILIDADE;
    LIMITER:=LIMITER;
    IF  REC OR LLI OR TR  THEN BEGIN
                                 SEQUENCIAVAZIA;
                                 START;
                               END;
    IF  REC OR TR  THEN
    BEGIN
      DETETARECURSAO;
      IF   RECURSAO
      THEN BEGIN
             LIMITER:=LIMITER;
             IF   GR THEN GRAMATICAFINAL(3)
             ELSE WRITE(OUTPUT,<///,10("*"),X10,"IMPRESSAO DA GRAM",
                          "ATICA APOS ELIMINACAO DA RECURSAO SUSPENS",
                          "A PELO USUARIO",X10,10("*"),>);
             SEQUENCIAVAZIA;
             FOR I:=1 STEP 1 UNTIL LIMITER DO
             FOR J:=1 STEP 1 UNTIL NI DO SREAL[I,J]:=0;
             START;
           END
      ELSE WRITE(OUTPUT,<///,10("*"),X10,"A GRAMATICA NAO E' RECUR",
                   "SIVA A ESQUERDA",X10,10("*"),>);
    END;
    IF  NOT FAT AND NOT REC  THEN A_FARETD;
    IF   (REC OR LLI OR TR) AND FAT  THEN OUTSTART
    ELSE WRITE(OUTPUT,<///,10("*"),X10,"IMPRESSAO DA MATRIZ DE SIM",
           "BOLOS INICIADORES SUSPENSA PELO USUARIO",X10,10("*"),>);
    IF  LLI OR TR  THEN FOLLOW;
    IF   (LLI OR TR) AND TAB  THEN OUTFOLLOW
    ELSE WRITE(OUTPUT,<///,10("*"),X10,"IMPRESSAO DA MATRIZ DE SIM",
           "BOLOS SEGUIDORES SUSPENSA PELO USUARIO",X10,10("*"),>);
    IF  LLI OR TR  THEN SINADIR;
  END
  ELSE WRITE(OUTPUT,<///,10("*"),X10,"CARTAO CONTROLE DO SISTEMA ",
           "INVALIDO",X10,10("*"),>);
  PARE:
 END;
END
END.
```

GRAMATICA LICA

OBS.: A NUMERACAO AO LADO DAS PRODUCOES CORRESPONDE AO NUMERO DO CARTAO LIDO

```
 1   PROGRAMA = BLOCO ;
 2   BLOCO    = 'BEGIN' IDENTIFICADORES*LOCAIS LISTA*DE*COMANDOS 'END' ;
 3   IDENTIFICADORES*LOCAIS = IDENTIFICADORES*LOCAIS 'LOCAL' IDENTIFICADOR ';'
 4                          |
 5   LISTA*DE*COMANDOS = COMANDO
 6                     | LISTA*DE*COMANDOS ';' COMANDO ;
 7   COMANDO = VARIAVEL ':=' EXPRESSAO
 8           | 'GOTO' IDENTIFICADOR
 9           | 'IF' EXPRESSAO 'THEN' COMANDO
10           | BLOCO
11           | 'RESULT' EXPRESSAO
12           | LABEL ':' COMANDO ;
13   EXPRESSAO = EXPRESSAO*ARITMETICA OPERADOR*RELACIONAL EXPRESSAO*ARITMETICA
14             | EXPRESSAO*ARITMETICA ;
15   EXPRESSAO*ARITMETICA = EXPRESSAO*ARITMETICA OPERADOR*DE*ADICAO TERMO
16                       | TERMO ;
17   TERMO = TERMO OPERADOR*DE*MULTIPLICACAO PRIMARIO
18         | PRIMARIO ;
19   PRIMARIO = VARIAVEL
20            | CONSTANTE
21            | '(' EXPRESSAO ')'
22            | BLOCO ;
23   VARIAVEL = IDENTIFICADOR
24            | IDENTIFICADOR '(' LISTA*DE*EXPRESSAO ')' ;
25   LISTA*DE*EXPRESSAO = LISTA*DE*EXPRESSAO ',' EXPRESSAO
26                      | EXPRESSAO ;
27   OPERADOR*RELACIONAL = '<' | '<=' | '=' | '<>' | '>' | '>=' ;
28   OPERADOR*DE*ADICAO = '+' | '-' ;
29   OPERADOR*DE*MULTIPLICACAO = '*' | '/' ;
30   LABEL = IDENTIFICADOR ;
31   IDENTIFICADOR = 'ID' ;
32   CONSTANTE = 'INT' ;
```

REPRESENTACAO INTERNA DA GRAMATICA

NUMERO DA PRODUCAO	LADO ESQUERDO	LADO DIREITO			
1	1	2			
2	2	201	3	4	202
3	3	3	203	5	204
4	3				
5	4	6			
6	4	4	204	6	
7	5	224			
8	6	7	205	8	
9	6	206	5		
10	6	207	8	208	6
11	6	2			
12	6	209	8		
13	6	9	210	6	
14	7	5			
15	7	5	211	17	212
16	8	10	11	10	
17	8	10			
18	9	5			
19	10	10	12	13	
20	10	13			
21	11	214			
22	11	215			
23	11	216			
24	11	217			
25	11	218			
26	11	219			
27	12	220			
28	12	221			
29	13	13	14	15	
30	13	15			
31	14	222			
32	14	223			
33	15	7			
34	15	16			
35	15	211	8	212	
36	15	2			
37	16	225			
38	17	17	213	6	
39	17	8			

NUMERO DE NAO-TERMINAIS = 17
NUMERO DE TERMINAIS = 25

GRAMATICA CODIFICADA

```
 1   1  PROGRAMA       =   BLOCO
 2   2  BLOCO    =   BEGIN        IDENTIFICADORES*LOCAIS         LISTA*DE*COMANDO         END
 3   3  IDENTIFICADORES*LOCAIS     =   IDENTIFICADORES*LOCAIS         LOCAL       IDENTIFICADOR
 4   4                              |
 5   5  LISTA*DE*COMANDO    =   COMANDO
 6   6                          |   LISTA*DE*COMANDO                    COMANDO
31   7  IDENTIFICADOR        =   ID
 7   8  COMANDO        =   VARIAVEL          :=          EXPRESSAO
 8   9                  |   GOTO          IDENTIFICADOR
 9  10                  |   IF            EXPRESSAO          THEN         COMANDO
10  11                  |   BLOCO
11  12                  |   RESULT        EXPRESSAO
12  13                  |   LABEL         :           COMANDO
23  14  VARIAVEL       =   IDENTIFICADOR
24  15                  |   IDENTIFICADOR                    LISTA*DE*EXPRESSAO       )
13  16  EXPRESSAO      =   EXPRESSAO*ARITMETICA          OPERADOR*RELACIONAL          EXPRESSAO*ARITMETICA
14  17                  |   EXPRESSAO*ARITMETICA
33  18  LABEL    =   IDENTIFICADOR
15  19  EXPRESSAO*ARITMETICA     =   EXPRESSAO*ARITMETICA          OPERADOR*DE*ACICAO       TERMO
16  20                           |   TERMO
27  21  OPERADOR*RELACIONAL      =   <
27  22                           |   <=
27  23                           |   =
27  24                           |   <>
27  25                           |   >
27  26                           |   >=
28  27  OPERADOR*DE*ADICAO  =   +
28  28                      |   -
17  29  TERMO      =   TERMO       OPERADOR*DE*MULTIPLICACAO          PRIMARIO
18  30          |   PRIMARIO
29  31  OPERADOR*DE*MULTIPLICACAO
29  32
19  33  PRIMARIO       =   VARIAVEL
20  34                 |   CONSTANTE
21  35                 |   (         EXPRESSAO
22  36                 |   BLOCO
```

GRAMATIÇA CODIFICADA

32	37	CONSTANTE	=	INT	
25	38	LISTA*DE*EXPRESSAO	=	LISTA*DE*EXPRESSAO	EXPRESSAO
26	39			EXPRESSAO	

OBS.: A NUMERACAO AO LADO DAS PRODUCOES CORRESPONDE
AO NUMERO DA CARTAO ONDE A PRODUCAO INICIA(PRIMEIRO NUMERO) E AO NUMERO DA PRODUCAO (SEGUNDO NUMERO)

GRAMATICA APOS FATORACAO

```
 1  PROGRAMA              =  BLOCO
 2  BLOCO                 =  BEGIN  IDENTIFICADORES*LOCAIS  LISTA*DE*COMANDO  END
 3  IDENTIFICADORES*LOCAIS =  IDENTIFICADORES*LOCAIS  LOCAL  IDENTIFICADOR
 4                         |
 5  LISTA*DE*COMANDO      =  COMANDO
 6                         |  LISTA*DE*COMANDO  COMANDO
 7  IDENTIFICADOR         =  ID
 8  COMANDO               =  VARIAVEL  :=  EXPRESSAO
 9                         |  GOTO  IDENTIFICADOR
10                         |  IF  EXPRESSAO  THEN  COMANDO
11                         |  BLOCO
12                         |  RESULT  EXPRESSAO
13                         |  LABEL  :  COMANDO
14  VARIAVEL              =  IDENTIFICADOR  NT18
15  EXPRESSAO             =  EXPRESSAO*ARITMETICA  NT19
16  LABEL                 =  IDENTIFICADOR
17  EXPRESSAO*ARITMETICA  =  EXPRESSAO*ARITMETICA  OPERADOR*DE*ADICAO  TERMO
18                         |  TERMO
19  OPERADOR*RELACIONAL   =  <
20                         |  <=
21                         |  =
22                         |  =>
23                         |  >
24                         |  >=
25  OPERADOR*DE*ADICAO    =  +
26                         |  -
27  TERMO                 =  TERMO  OPERADOR*DE*MULTIPLICACAO  PRIMARIO
28                         |  PRIMARIO
29  OPERADOR*DE*MULTIPLICACAO  =  *
30                             |  /
31  PRIMARIO              =  VARIAVEL
32                         |  CONSTANTE
33                         |  (  EXPRESSAO
34                         |  BLOCO
35  CONSTANTE             =  INT
36  LISTA*DE*EXPRESSAO    =  LISTA*DE*EXPRESSAO  EXPRESSAO
```

GRAMÁTICA APÓS FATORAÇÃO

```
37                      |   EXPRESSAO
38      NT16    =
39              |   (         LISTA*DE*EXPRESSAO
40      NT19    =   OPERADOR*RELACIONAL         EXPRESSAO*ARITMETICA
41              |
```

OBS.: A NUMERAÇÃO AO LADO DAS PRODUÇÕES CORRESPONDE AO NÚMERO DA PRODUÇÃO

********** NAO-TERMINAIS QUE GERAM SEQUENCIA VAZIA **********

IDENTIFICADORES*LOCAIS

NT18

NT19

********** GRUPO DE NAO-TERMINAIS RECURSIVOS **********

IDENTIFICADORES*LOCAIS
LISTA*DE*COMANDO
EXPRESSAO*ARITMETICA
TERMO
LISTA*DE*EXPRESSAO

```
*****************************************************************
*                                                               *
*                GRAMATICA APOS ELIMINACAO DA RECURSAO          *
*                                                               *
*****************************************************************

 1   PROGRAMA          =  BLOCO
 2   BLOCO     =   BEGIN      IDENTIFICADORES*LOCAIS        LISTA*DE*COMANDO        END.
 3   IDENTIFICADORES*LOCAIS     =   NT20
 4   LISTA*DE*COMANDO     =   COMANDO            NT21
 5   IDENTIFICADOR        =   IC
 6   COMANDO        =   VARIAVEL          :=          EXPRESSAO
 7                  |   GOTO        IDENTIFICADOR
 8                  |   IF          EXPRESSAO       THEN        COMANDO
 9                  |   BLOCO
10                  |   RESULT      EXPRESSAO
11                  |   LABEL       :           COMANDO
12   VARIAVEL       =   IDENTIFICADOR           NT16
13   EXPRESSAO      =   EXPRESSAO*ARITMETICA          NT19
14   LABEL     =   IDENTIFICADOR
15   EXPRESSAO*ARITMETICA       =   TERMO       NT22
16   OPERADOR*RELACIONAL        =   <
17                              |   <=
18                              |   =
19                              |   =>
20                              |   >
21                              |   >=
22   OPERADOR*DE*ADICAO    =   +
23                         |   -
24   TERMO     =   PRIMARIO          NT23
25   OPERADOR*DE*MULTIPLICACAO        =   *
26                                    |   /
27   PRIMARIO       =   VARIAVEL
28                  |   CONSTANTE
29                  |   (           EXPRESSAO
30                  |   BLOCO
31   CONSTANTE      =   INT
32   LISTA*DE*EXPRESSAO    =   EXPRESSAO          NT24
33   NT18      =
34             |   (           LISTA*DE*EXPRESSAO       )
35   NT19      =   OPERADOR*RELACIONAL-          EXPRESSAO*ARITMETICA
36             |
```

GRAMATICA APOS ELIMINACAO DA RECURSAO

37	NT20	=	LOCAL	IDENTIFICADOR	;	NT20	
38		|					
39	NT21	=	;	COMANDO	NT21		
40		|					
41	NT22	=	OPERADOR*DE*ADICAO	TERMO	NT22		
42		|					
43	NT23	=	OPERADOR*DE*MULTIPLICACAO		PRIMARIO	NT23	
44		|					
45	NT24	=	,	EXPRESSAO	NT24		
46		|					

OBS.: A NUMERACAO AO LADO DAS PRODUCOES CORRESPONDE AO NUMERO DA PRODUCAO

********** NAO-TERMINAIS QUE GERAM SEQUENCIA VAZIA **********

NT18
NT19
NT20
NT21
NT22
NT23
NT24

CODIGO DO ALFABETO

1	PROGRAMA
2	BLOCO
3	IDENTIFICADORES*LOCAIS
4	LISTA*DE*COMANDO
5	IDENTIFICADOR
6	COMANDO
7	VARIAVEL
8	EXPRESSAO
9	LABEL
10	EXPRESSAO*ARITMETICA
11	OPERADOR*RELACIONAL
12	OPERADOR*DE*ADICAO
13	TERMO
14	OPERADOR*DE*MULTIPLICACAO
15	PRIMARIO
16	CONSTANTE
17	LISTA*DE*EXPRESSAO
18	NT18
19	NT19
20	NT20
21	NT21
22	NT22
23	NT23
24	NT24
25	BEGIN
26	END
27	LOCAL
28	;
29	:=
30	GOTO
31	IF
32	THEN
33	RESULT
34	:
35	(
36	)
37	,
38	<
39	<=
40	=
41	<>
42	>
43	>=
44	+
45	-
46	*
47	/
48	ID
49	INT

MATRIZ DE SIMBOLOS INICIADORES

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	1	0	1	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	1	0	0	1	0	1	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MATRIZ DE SIMBOLOS INICIADORES

	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
6	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
8	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
13	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
15	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
17	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
18	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0
20	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

MATRIZ DE SIMBOLOS SEGUIDORES

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MATRIZ DE SIMBOLOS SEGUIDORES

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	1	0	1	1	1	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0	0
6	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0	0	0	0	0
8	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	1	0	1	0	1	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0
16	0	0	0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	1	0	1	1	1	0	1	0	0	1	0	1	0	1	0	1	0	0	0	0	0
19	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
20	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0
23	0	0	0	1	0	1	0	1	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

MATRIZ DE SIMBOLOS SEGUIDORES

	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	1	0	1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1
3	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	1	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
6	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	1	0	1	1	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
8	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	1	0	1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
11	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
12	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
13	0	1	0	1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
14	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
15	0	1	0	1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
16	0	1	0	1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	1	0	1	1	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
19	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
20	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
21	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	1	0	1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
23	0	1	0	1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CONJUNTO DE SIMBOLOS DIRETORES - NAO-TERMINAL ==> (TERMINAL,NUMERO DA PRODUCAO)

```
 1 ==> ( 25 ,  1 )
 2 ==> ( 25 ,  2 )
 3 ==> ( 27 ,  3 )
 4 ==> ( 25 ,  4 ) ( 30 ,  4 ) ( 31 ,  4 ) ( 33 ,  4 ) ( 48 ,  4 )
 5 ==> ( 48 ,  5 )
 6 ==> ( 48 , 11 ) ( 33 , 11 ) ( 25 ,  9 ) ( 31 ,  8 ) ( 30 ,  7 ) ( 48 ,  6 )
 7 ==> ( 48 , 12 )
 8 ==> ( 25 , 13 ) ( 35 , 13 ) ( 48 , 13 ) ( 49 , 13 )
 9 ==> ( 48 , 14 )
10 ==> ( 25 , 15 ) ( 35 , 15 ) ( 48 , 15 ) ( 49 , 15 )
11 ==> ( 43 , 21 ) ( 42 , 20 ) ( 41 , 19 ) ( 40 , 18 ) ( 39 , 17 ) ( 38 , 16 )
12 ==> ( 45 , 23 ) ( 44 , 22 )
13 ==> ( 25 , 24 ) ( 35 , 24 ) ( 48 , 24 ) ( 49 , 24 )
14 ==> ( 47 , 26 ) ( 46 , 25 )
15 ==> ( 25 , 30 ) ( 35 , 29 ) ( 49 , 28 ) ( 48 , 27 )
16 ==> ( 49 , 31 )
17 ==> ( 25 , 32 ) ( 35 , 32 ) ( 48 , 32 ) ( 49 , 32 )
18 ==> ( 35 , 34 ) ( 26 , 33 ) ( 28 , 33 ) ( 29 , 33 ) ( 32 , 33 ) ( 36 , 33 ) ( 37 , 33 ) ( 38 , 33 )
       ( 39 , 33 ) ( 40 , 33 ) ( 41 , 33 ) ( 42 , 33 ) ( 43 , 33 ) ( 44 , 33 ) ( 45 , 33 ) ( 46 , 33 )
       ( 47 , 33 )
19 ==> ( 26 , 36 ) ( 28 , 36 ) ( 32 , 36 ) ( 36 , 36 ) ( 37 , 36 ) ( 38 , 35 ) ( 39 , 35 ) ( 40 , 35 )
       ( 41 , 35 ) ( 42 , 35 ) ( 43 , 35 )
20 ==> ( 25 , 38 ) ( 30 , 38 ) ( 31 , 38 ) ( 33 , 38 ) ( 48 , 38 ) ( 27 , 37 )
21 ==> ( 26 , 40 ) ( 28 , 39 )
22 ==> ( 26 , 42 ) ( 28 , 42 ) ( 32 , 42 ) ( 36 , 42 ) ( 37 , 42 ) ( 38 , 42 ) ( 39 , 42 ) ( 40 , 42 )
       ( 41 , 42 ) ( 42 , 42 ) ( 43 , 42 ) ( 44 , 41 ) ( 45 , 41 )
23 ==> ( 26 , 44 ) ( 28 , 44 ) ( 32 , 44 ) ( 36 , 44 ) ( 37 , 44 ) ( 38 , 44 ) ( 39 , 44 ) ( 40 , 44 )
       ( 41 , 44 ) ( 42 , 44 ) ( 43 , 44 ) ( 44 , 44 ) ( 45 , 44 ) ( 46 , 43 ) ( 47 , 43 )
24 ==> ( 36 , 46 ) ( 37 , 45 )
```

A GRAMATICA NAO E' LL(1) EM DECORRENCIA DOS CONFLITOS ABAIXO

(NAO-TERMINAL,TERMINAL)<=>PRODUCOES
(6 , 48) => 11 , 6

```
**********************************************************************
*                                                                    *
*                          GRAMATICA LIDA                            *
*                                                                    *
**********************************************************************
```

OBS.: A NUMERACAO AO LADO DAS PRODUCOES CORRESPONDE AO NUMERO DO CARTAO LIDO

```
 1    PROGRAMA  =  BLOCO  ;
 2    BLOCO     =  'BEGIN'  IDENTIFICADORES*LOCAIS  LISTA*DE*COMANDO  'END'  ;
 3    IDENTIFICADORES*LOCAIS  =  IDENTIFICADORES*LOCAIS  'LOCAL'  IDENTIFICADOR  ';'
 4                            |
 5    LISTA*DE*COMANDO  =  COMANDO
 6                      |  LISTA*DE*COMANDO  ';'  COMANDO
 7    COMANDO  =  'GOTO'  IDENTIFICADOR
 8             |  'IF'  EXPRESSAO  'THEN'  COMANDO
 9             |  BLOCO
10             |  'RESULT'  EXPRESSAO  ;
11    EXPRESSAO  =  EXPRESSAO*ARITMETICA  OPERADOR*RELACIONAL  EXPRESSAO*ARITMETICA
12               |  EXPRESSAO*ARITMETICA  ;
13    EXPRESSAO*ARITMETICA  =  EXPRESSAO*ARITMETICA  OPERADOR*DE*ADICAO  TERMO
14                          |  TERMO  ;
15    TERMO  =  TERMO  OPERADOR*DE*MULTIPLICACAO  PRIMARIO
16           |  PRIMARIO  ;
17    PRIMARIO  =  VARIAVEL
18              |  CONSTANTE
19              |  '('  EXPRESSAO  ')'
20              |  BLOCO  ;
21    VARIAVEL  =  IDENTIFICADOR
22              |  IDENTIFICADOR  '('  LISTA*DE*EXPRESSAO  ')'  ;
23    LISTA*DE*EXPRESSAO  =  LISTA*DE*EXPRESSAO  ','  EXPRESSAO
24                        |  EXPRESSAO  ;
25    OPERADOR*RELACIONAL  =  '<'  |  '<='  |  '='  |  '=='  |  '>'  |  '>='  ;
26    OPERADOR*DE*ADICAO  =  '+'  |  '-'  ;
27    OPERADOR*DE*MULTIPLICACAO  =  '*'  |  '/'  ;
28    IDENTIFICADOR  =  'ID'  ;
29    CONSTANTE  =  'INT'  ;
30    COMANDO  =  IDENTIFICADOR  ':='  EXPRESSAO
31             |  IDENTIFICADOR  '('  LISTA*DE*EXPRESSAO  ')'  ':='  EXPRESSAO
32             |  IDENTIFICADOR  ':'  COMANDO  ;
```

REPRESENTACAO INTERNA DA GRAMATICA

NUMERO DA PRODUCAO	LADO ESQUERDO	LADO DIREITO				
1	1	2				
2	2	211	3	4	202	
3	3	3	203	5	204	
4	4	6				
5	4	4	204	6		
6	5	222				
7	5	225	5			
8	6	206	7	207	6	
9	6	2				
10	6	208	7			
11	6	5	224	7		
12	6	5	209	16	210	224 7
13	6	225	6			
14	7	8	9	8		
15	7	8				
16	8	8	10	11		
17	8	11				
18	9	212				
19	9	213				
20	9	214				
21	9	215				
22	9	216				
23	9	217				
24	10	218				
25	10	219				
26	11	11	12	13		
27	11	13				
28	12	220				
29	12	221				
30	13	16				
31	13	15				
32	13	209	7	210		
33	13	2				
34	14	5				
35	14	6	209	16	210	
36	15	224				
37	16	16	211	7		
38	16	7				

NUMERO DE NAO-TERMINAIS = 16
NUMERO DE TERMINAIS = 25

GRAMATICA CODIFICADA

```
 1    1  PROGRAMA      =  BLOCO
 2    2  BLOCO     =  BEGIN       IDENTIFICADORES*LOCAIS        LISTA*DE*COMANDO        END
 3    3  IDENTIFICADORES*LOCAIS     =  IDENTIFICADORES*LOCAIS      LOCAL       IDENTIFICADOR
 4    4                             |
 5    5  LISTA*DE*COMANDO     =  COMANDO
 6    6                          |  LISTA*DE*COMANDO                    COMANDO
28    7  IDENTIFICADOR         =  IC
 7    8  COMANDO         =  GOTO         IDENTIFICADOR
 8    9                    |  IF          EXPRESSAO          THEN        COMANDO
 9   10                    |  BLOCO
10   11                    |  RESULT      EXPRESSAO
30   12                    |  IDENTIFICADOR            :=          EXPRESSAO
31   13                    |  IDENTIFICADOR            (           LISTA*DE*EXPRESSAO                    :=          EXPRESSAO
32   14                    |  IDENTIFICADOR            :           COMANDO
11   15  EXPRESSAO      =  EXPRESSAO*ARITMETICA        OPERADOR*RELACIONAL          EXPRESSAO*ARITMETICA
12   16                    |  EXPRESSAO*ARITMETICA
13   17  EXPRESSAO*ARITMETICA      =  EXPRESSAO*ARITMETICA        OPERADOR*DE*ADICAO      TERMO
14   18                             |  TERMO
25   19  OPERADOR*RELACIONAL       =  <
25   20                             |  <=
25   21                             |  =
25   22                             |  <>
25   23                             |  >
25   24                             |  >=
26   25  OPERADOR*DE*ADICAO  =  +
26   26                        |  -
15   27  TERMO     =  TERMO       OPERADOR*DE*MULTIPLICACAO          PRIMARIO
16   28             |  PRIMARIO
27   29  OPERADOR*DE*MULTIPLICACAO       =  *
27   30                                   |  /
17   31  PRIMARIO       =  VARIAVEL
18   32                 |  CONSTANTE
19   33                 |  (          EXPRESSAO
20   34                 |  BLOCO
21   35  VARIAVEL       =  IDENTIFICADOR
22   36                 |  IDENTIFICADOR                    LISTA*DE*EXPRESSAO
```

GRAMATICA CODIFICADA

29	37	CONSTANTE	=	INT	
23	38	LISTA*DE*EXPRESSAO	=	LISTA*DE*EXPRESSAO	EXPRESSAO.
24	39			EXPRESSAO	

OBS.: A NUMERACAO AO LADO DAS PRODUCOES CORRESPONDE
AO NUMERO DA CARTAO ONDE A PRODUCAO INICIA(PRIMEIRO NUMERO) E AO NUMERO DA PRODUCAO (SEGUNDO NUMERO)

```
*****************************************************************************
*                                                                           *
*                          GRAMATICA APOS FATORACAO                         *
*                                                                           *
*****************************************************************************

 1   PROGRAMA          =   BLOCO
 2   BLOCO             =   BEGIN   IDENTIFICADORES*LOCAIS   LISTA*DE*COMANDOS   END
 3   IDENTIFICADORES*LOCAIS  =   IDENTIFICADORES*LOCAIS   LOCAL   IDENTIFICADOR
 4                           |
 5   LISTA*DE*COMANDOS     =   COMANDO
 6                           |   LISTA*DE*COMANDOS   ;   COMANDO
 7   IDENTIFICADOR         =   IC
 8   COMANDO           =   GOTO    IDENTIFICADOR
 9                     |   IF      EXPRESSAO   THEN   COMANDO
10                     |   BLOCO
11                     |   RESULT  EXPRESSAO
12                     |   IDENTIFICADOR   NT18
13   EXPRESSAO         =   EXPRESSAO*ARITMETICA   NT19
14   EXPRESSAO*ARITMETICA  =   EXPRESSAO*ARITMETICA   OPERADOR*DE*ADICAO   TERMO
15                         |   TERMO
16   OPERADOR*RELACIONAL   =   <
17                         |   <=
18                         |   =
19                         |   <>
20                         |   >
21                         |   >=
22   OPERADOR*DE*ADICAO    =   +
23                         |   -
24   TERMO     =   TERMO   OPERADOR*DE*MULTIPLICACAO   PRIMARIO
25             |   PRIMARIO
26   OPERADOR*DE*MULTIPLICACAO   =   *
27                               |   /
28   PRIMARIO          =   VARIAVEL
29                     |   CONSTANTE
30                     |   (   EXPRESSAO
31                     |   BLOCO
32   VARIAVEL          =   IDENTIFICADOR   NT20
33   CONSTANTE         =   INT
34   LISTA*DE*EXPRESSAO    =   LISTA*DE*EXPRESSAO   EXPRESSAO
35                         |   EXPRESSAO
36   NT17      =   :=   EXPRESSAO
```

GRAMATICA APOS FATORACAO

37		I	(LISTA*DE*EXPRESSAO)	EXPRESSAO
38	NT18	=	NT17	
39		I	: COMANDO	
40	NT19	=	OPERADOR*RELACIONAL EXPRESSAO*ARITMETICA	
41		I		
42	NT20	=		
43		I	(LISTA*DE*EXPRESSAO)	

OBS.: A NUMERACAO AO LADO DAS PRODUCOES CORRESPONDE AO NUMERO D- PRODUCAO

********** NAO-TERMINAIS QUE GERAM SEQUENCIA VAZIA **********

IDENTIFICADORES*LOCAIS
NT19
NT20

********** GRUPO DE NAO-TERMINAIS RECURSIVOS **********

IDENTIFICADORES*LOCAIS
LISTA*DE*COMANDO
EXPRESSAO*ARITMETICA
TERMO
LISTA*DE*EXPRESSAO

GRAMATICA APOS ELIMINACAO DA RECURSAO

```
1    PROGRAMA          =   BLOCO
2    BLOCO    =   BEGIN        IDENTIFICADORES*LOCAIS        LISTA*DE*COMANDO        END
3    IDENTIFICADORES*LOCAIS     =   NT21
4    LISTA*DE*COMANDO  .  =   COMANDO            NT22
5    IDENTIFICADOR          =   ID
6    COMANDO          =   GOTO          IDENTIFICADOR
7                     |   IF            EXPRESSAO          THEN          COMANDO
8                     |   BLOCO
9                     |   RESULT        EXPRESSAO
10                    |   IDENTIFICADOR            NT18
11   EXPRESSAO        =   EXPRESSAO*ARITMETICA          NT19
12   EXPRESSAO*ARITMETICA        =   TERMO         NT23
13   OPERADOR*RELACIONAL         =   <
14                               |   <=
15                               |   =
16                               |   $=
17                               |   >
18                               |   >=
19   OPERADOR*DE*ADICAO  =   +
20                       |   -
21   TERMO    =   PRIMARIO            NT24
22   OPERADOR*DE*MULTIPLICACAO        =   *
23                                    |   /
24   PRIMARIO         =   VARIAVEL
25                    |   CONSTANTE
26                    |   (           EXPRESSAO )
27                    |   BLOCO
28   VARIAVEL         =   IDENTIFICADOR           NT20
29   CONSTANTE        =   INT
30   LISTA*DE*EXPRESSAO  =   EXPRESSAO          NT25
31   NT17     =   :=          EXPRESSAO
32            |   (           LISTA*DE*EXPRESSAO )          :=          EXPRESSAO
33   NT18     =   NT17
34            |   :           COMANDO
35   NT19     =   OPERADOR*RELACIONAL          EXPRESSAO*ARITMETICA
36
```

GRAMATICA APOS ELIMINACAO DA RECURSAO

```
37      NT20    =
38              |   (             LISTA*DE*EXPRESSAO
39      NT21    =   LOCAL         IDENTIFICADOR                       NT21
40              |
41      NT22    =   ;             COMANDO             NT22
42              |
43      NT23    =   OPERADOR*DE*ADICAO      TERMO           NT23
44              |
45      NT24    =   OPERADOR*DE*MULTIPLICACAO               PRIMARIO            NT24
46              |
47      NT25    =   ,             EXPRESSAO           NT25
48              |
```

OBS.: A NUMERACAO AO LADO DAS PRODUCOES CORRESPONDE AO NUMERO DA PRODUCAO

********** NAO-TERMINAIS QUE GERAM SEQUENCIA VAZIA **********

NT19
NT20
NT21
NT22
NT23
NT24
NT25

CODIGO DO ALFABETO

1	PROGRAMA
2	BLOCO
3	IDENTIFICADORES*LOCAIS
4	LISTA*DE*COMANDO
5	IDENTIFICADOR
6	COMANDO
7	EXPRESSAO
8	EXPRESSAO*ARITMETICA
9	OPERADOR*RELACIONAL
10	OPERADOR*DE*ADICAO
11	TERMO
12	OPERADOR*DE*MULTIPLICACAO
13	PRIMARIO
14	VARIAVEL
15	CONSTANTE
16	LISTA*DE*EXPRESSAO
17	NT17
18	NT18
19	NT19
20	NT20
21	NT21
22	NT22
23	NT23
24	NT24
25	NT25
26	BEGIN
27	END
28	LOCAL
29	;
30	GOTO
31	IF
32	THEN
33	RESULT
34	(
35	)
36	,
37	<
38	<=
39	=
40	<>
41	>
42	>=
43	+
44	-
45	*
46	/
47	ID
48	INT
49	:=
50	:

MATRIZ DE SIMBOLOS INICIADORES

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
4	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	1	0	0	1	0	0	1	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	1	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	1	0	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	1	0	0	1	0	1	1	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MATRIZ DE SIMBOLOS INICIADORES

	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
6	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
7	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
8	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
11	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
13	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
16	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
17	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
18	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
19	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MATRIZ DE SIMBOLOS SEGUIDORES

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MATRIZ DE SIMBOLOS SEGUIDORES

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	1	1	1	0	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
7	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
8	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	1	0	1	1	1	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0
14	0	0	0	1	0	1	1	1	0	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0
15	0	0	0	1	0	1	1	1	0	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
18	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
19	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
20	0	0	0	1	0	1	1	1	0	0	1	0	1	1	0	1	1	1	1	0	0	0	0	0	0
21	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
24	0	0	0	1	0	1	1	1	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

MATRIZ DE SIMBOLOS SEGUIDORES

.	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1
3	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
6	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	1	0	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
10	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
11	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
12	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
13	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
14	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
15	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	1	0	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
21	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
22	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
24	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CONJUNTO DE SIMBOLOS DIRETORES - NAO-TERMINAL ==> (TERMINAL,NUMERO DA PRODUCAO)

```
 1 ==> ( 26 ,  1 )
 2 ==> ( 26 ,  2 )
 3 ==> ( 28 ,  3 )
 4 ==> ( 26 ,  4 ) ( 30 ,  4 ) ( 31 ,  4 ) ( 33 ,  4 ) ( 47 ,  4 )
 5 ==> ( 47 ,  5 )
 6 ==> ( 47 , 10 ) ( 33 ,  9 ) ( 26 ,  8 ) ( 31 ,  7 ) ( 30 ,  6 )
 7 ==> ( 26 , 11 ) ( 34 , 11 ) ( 47 , 11 ) ( 48 , 11 )
 8 ==> ( 26 , 12 ) ( 34 , 12 ) ( 47 , 12 ) ( 48 , 12 )
 9 ==> ( 42 , 18 ) ( 41 , 17 ) ( 40 , 16 ) ( 39 , 15 ) ( 38 , 14 ) ( 37 , 13 )
10 ==> ( 44 , 20 ) ( 43 , 19 )
11 ==> ( 26 , 21 ) ( 34 , 21 ) ( 47 , 21 ) ( 48 , 21 )
12 ==> ( 46 , 23 ) ( 45 , 22 )
13 ==> ( 26 , 27 ) ( 34 , 26 ) ( 48 , 25 ) ( 47 , 24 )
14 ==> ( 47 , 28 )
15 ==> ( 48 , 29 )
16 ==> ( 26 , 30 ) ( 34 , 30 ) ( 47 , 30 ) ( 48 , 30 )
17 ==> ( 34 , 32 ) ( 49 , 31 )
18 ==> ( 50 , 34 ) ( 34 , 33 ) ( 49 , 33 )
19 ==> ( 27 , 36 ) ( 29 , 36 ) ( 32 , 36 ) ( 35 , 36 ) ( 36 , 36 ) ( 37 , 35 ) ( 38 , 35 ) ( 39 , 35 )
       ( 40 , 35 ) ( 41 , 35 ) ( 42 , 35 )
20 ==> ( 34 , 38 ) ( 27 , 37 ) ( 29 , 37 ) ( 32 , 37 ) ( 35 , 37 ) ( 36 , 37 ) ( 37 , 37 ) ( 38 , 37 )
       ( 39 , 37 ) ( 40 , 37 ) ( 41 , 37 ) ( 42 , 37 ) ( 43 , 37 ) ( 44 , 37 ) ( 45 , 37 ) ( 46 , 37 )
21 ==> ( 26 , 40 ) ( 30 , 40 ) ( 31 , 40 ) ( 33 , 40 ) ( 47 , 40 ) ( 28 , 39 )
22 ==> ( 27 , 42 ) ( 29 , 41 )
23 ==> ( 27 , 44 ) ( 29 , 44 ) ( 32 , 44 ) ( 35 , 44 ) ( 36 , 44 ) ( 37 , 44 ) ( 38 , 44 ) ( 39 , 44 )
       ( 40 , 44 ) ( 41 , 44 ) ( 42 , 44 ) ( 43 , 43 ) ( 44 , 43 )
24 ==> ( 27 , 46 ) ( 29 , 46 ) ( 32 , 46 ) ( 35 , 46 ) ( 36 , 46 ) ( 37 , 46 ) ( 38 , 46 ) ( 39 , 46 )
       ( 40 , 46 ) ( 41 , 46 ) ( 42 , 46 ) ( 43 , 46 ) ( 44 , 46 ) ( 45 , 45 ) ( 46 , 45 )
25 ==> ( 35 , 48 ) ( 36 , 47 )
```

A GRAMATICA E' LL(1)