I

UM GERADOR DE TABELAS DE ANÁLISE SINTÁTICA PARA GRAMÁTICAS SLR(1), LALR(1) e LR(1)

RAUL VICENTE TARDIN COSTA

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 Mauad Rangel Netto
Presidente

Luis Alberto

Ebert Schmitz

RIO DE JANEIRO, RJ- BRASIL
MARÇO DE 1981

COSTA, RAUL V. TARDIN

Um Gerador de Tabelas de Análise Sintática para Gramáticas SLR(1), LALR(1) e LR(1) (Rio de Janeiro 1981).

VIII, 243p. 29,7cm (COPPE-UFRJ, M.Sc. Sistemas, 1981).

Tese- Univ.Fed.Rio de Janeiro. Fac.Engenharia

1. Tabelas de análise sintática para Gramáticas SLR(1), LALR(1) e LR(1). I.COPPE/UFRJ.II.Título (série)

AGRADECIMENTOS

A JOSÉ LUCAS RANGEL pela orientação de todo o trabalho, sem a qual seria impossível realizá-lo.

A JAYME LUIZ SCWARCFITER e EBER ASSIS SCHMITZ, pela participação na banca examinadora.

Muito Obrigado.

RESUMO

Este trabalho implementa um gerador de tabelas para análise sintática de gramáticas SLR(1), LALR(1) e LR(1). A entrada é a gramática codificada na notação BNF "Backus Naur Form".

O gerador foi programado em linguagem ALGOL e implementado no computador Burroughs B6700 no Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro.

São apresentados : introdução teórica, algorítmos, características de implementação, documentação do programa, forma de utilização com uma série de exemplos e conclusões finais.

ABSTRACT

This work implements a parser table generator for gramars SLR(1), LALR(1) and LR(1).

The input is the grammar coded in BNF (Backus Naur Form) notation.

The generator was programmed in ALGOL language and was implemented on the Burroughs B6700 Computer of the Nucleo de Computação Eletrônica (Computing Center) of the Universidade Federal do Rio de Janeiro (Federal University of Rio de Janeiro).

The following subjects are presented : introduction, theoretical foundations, algorithms, implementation characteristics , program documentation, utilization form with a series of examples and final conclusions.

ÍNDICE

III. IMPLEMENTAÇÃO

IV. DOCUMENTAÇÃO

I. INTRODUÇÃO

1. Escopo

O analisador sintático de tipo LR é uma excelente opção na escolha de analisador sintático para uma gramática de uma dada linguagem por uma série de razões:

- A deteção de erros ocorre com maior antecipação possível em um percurso da esquerda para a direita de uma sentença de entrada. A tabela do analisador sintático contém os símbolos que seriam continuações válidas, facilitando as mensagens e recuperação de erro.

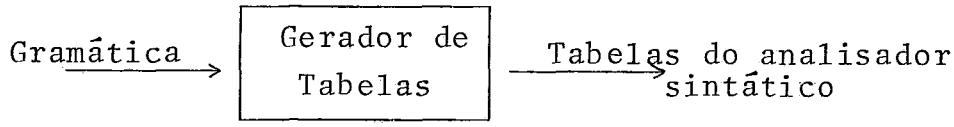
- O método de análise sintática LR é mais geral que o método de operadores de precedência e outras técnicas "shift-reduce". O método LR também domina as formas mais comuns de análise sintática "top-down" sem "back track" (AHO ULLMAN[1]).

- Estudos mostraram que os analisadores sintáticos LR podem ser construídos tão eficientemente quanto qualquer outra técnica de generalidade comparável , tanto em espaço como em tempo. Comparando as demandas de outros processos dentro de uma compilação, os requerimentos dos analisadores sintáticos LR são relativamente modestos (HORNING[2]).

Porém o que normalmente recebe um peso razoavelmente alto em uma escolha de analisador sintático, é a existência de gerador capaz de montar as tabelas deste, evitando assim a tarefa cansativa e demorada de montagem manual.

O objetivo desta tese é tornar disponível um gerador das tabelas de análise sintática para gramáticas do tipo LR, eliminando assim uma desvantagem da escolha de um analisador sintático LR.

A figura a seguir mostra o esboço geral do gerador de tabelas.



Geração de Tabelas

Este trabalho e uma série de outros foram sugeridos pelo grupo de professores da Área de Compiladores, entre os quais o orientador desta tese, de forma a montar estrutura capaz de dar suporte a transferência de "know-how" na construção de compiladores nacionais, deixando para o construtor apenas tarefas que dependam da interface com a máquina em que o compilador está sendo implementado. Um exemplo típico destas tarefas é a geração de código normalmente no Assembler da máquina.

2. Revisão da Literatura

A análise sintática LR foi inicialmente descrita em (KNUTH[6]).

As técnicas de análise sintática LR não vinham sendo utilizadas com a intensidade de outros métodos teoricamente menos atrativos. As apresentações originais do método eram de difícil compreensão e as implementações a partir do método original de KNUTH eram muito ineficientes.

Uma série de otimizações e apresentações mais claras foram descobertas (AHO ULLMAN[4]) (AHO ULLMAN[5]) (ANDERSON[7]) (DE REMER[8]) (JOLIAT[9]) (PAGER[10]) (HORNING[2]).

No início do trabalho desta Tese tomou-se conhecimento do livro (AHO ULLMAN[1]) que explica de forma bastante clara o algorítmo padrão de análise sintática LR, os algorítmos de tabelas de análise sintática SLR(1), LALR(1) e LR(1). Esta referência passou a ser a principal da implementação nesta tese realizada.

II. FUNDAMENTOS TEÓRICOS

Este capítulo revê rapidamente conceitos e definições necessários para utilização do produto desta Tese. Para maior detalhamento deve-se recorrer a bibliografia (AHO ULLMAN[1]) (HORNING[2]) (GRIES[3]) (AHO ULLMAN[4]) (AHO ULLMAN[5]).

A maior parte deste capítulo segue as referências (AHO ULLMAN[1]) (AHO ULLMAN[4]) (AHO ULLMAN[5]). A notação utilizada é baseada na convenção da referência (AHO ULLMAN[1]).

1. Convenções

1.1- Símbolos usualmente não terminais:

- nomes cuja primeira letra é maíscula tais como Expressão, Termo;
- letras maísculas;
- a letra S quando aparece é normalmente o símbolo inicial;
- nomes entre os caracteres < e > tais como < expressão > , < Expressão > , < E > , etc.

1.2- Símbolo usualmente terminais:

- letra minúscula perto do início do alfabeto a,b, c,.... ;
- operadores tais como +, - , / , etc ;
- caracteres de pontuação tais como parêntesis, vírgula, ponto, etc...;
- dígitos ou cadeia de dígitos tais como 0, 10, 2,245 etc...;
- nomes minúsculos tais como identificador, constante, etc....

1.3- Letras minúsculas perto do fim do alfabeto principalmente u, v, ..., z, representam cadeias de terminais.

1.4- Letras Gregas tais como α, β, γ representam cadeias de símbolos da gramática.

1.5- O lado esquerdo da primeira produção de uma gramática é normalmente o símbolo inicial.

1.6- Letras maiúsculas próximas ao fim do alfabeto, principalmente X, Y, Z, representam símbolos da gramática, isto é, terminais ou não terminais.

2. Gramática livre de contexto

Para a especificação de uma linguagem de programação utiliza-se normalmente de uma definição inicial de uma gramática livre de contexto descrita na forma conhecida como BNF "Backus-Naur Form".

Uma gramática livre de contexto é a gramática $G=(N,\Sigma,P,S)$ onde as produções de P são da forma :

$A::= \alpha$ onde $A \in N$ e $\alpha \in (N \cup \Sigma)^*$

e Σ - conjunto de símbolos terminais

N - conjunto de símbolos não terminais

S - símbolo inicial

A linguagem desta gramática é o conjunto de cadeias de terminais que podem ser gerados a partir de S de Σ^*.

As vantagens de se utilizar de uma gramática livre de contexto para definir sintaticamente uma linguagem de programação a esta associada são as seguintes :

- A gramática fornece uma forma fácil de entendimento da especificação sintática dos programas desta linguagem de programação.

- Podem ser construídos analisadores sintáticos automáticos para uma gramática bem desenhada, que no caso desta tese são os de tipo SLR(1), LALR(1) e LR(1). Portanto um requisito de utilização desta tese é a descrição da linguagem por uma gramática livre de contexto na notação BNF.

- A gramática cria uma estrutura para metodização da geração de código e detecção de erros.

3. Análise Sintática

Analisar sintaticamente uma sentença de uma linguagem de uma certa gramática consiste em verificar se esta sentença pertence a esta linguagem (caso em que é possível obter a árvore sintática das derivações realizadas para produzí-la) ou descobre-se a existência de erro que indica que a sentença não pertence a linguagem. Os métodos modernos de análise sintática estão montados normalmente em tabelas as quais podem ser anexados facilmente procedimentos de geração de código e recuperação de erro.

4. Métodos mais comuns de análise sintática

Existem dois tipos básicos de analisador sintático; o "top-down" que constroi a árvore sintática a partir do topo (raiz) e trabalha até a parte de baixo (folhas), e o "bottom - up" que trabalha na forma inversa. Nos dois casos a análise sintática considerada é a que percorre a sentença a reconhecer da esquerda para a direita que é como estamos acostumados a ler.

4.1- Métodos "bottom-up" :

Os métodos "bottom-up" mais conhecidos utilizam técnica conhecida como "shift-reduce" que consiste empilhar símbolos reconhecidos na sentença de entrada até aparecer o lado direito de alguma produção no topo da pilha, e este lado direito é então reduzido para o não terminal do lado esquerdo da produção. O processo continua da mesma maneira até o empilhamento do último símbolo e reduções até a produção que contém o símbolo inicial da gramática.

Definições:

Seja G (S) uma gramática e $\gamma = \alpha\beta w$ uma forma sentencial a direita, chama-se "handle" de γ à produção $A \to \beta$ e a posição de γ onde β pode ser encontrado e substituído por A, de forma a produzir a forma sentencial a di -

reita imediatamente anterior em uma derivação mais a direita de γ. Isto é, se $S \overset{*}{\underset{md}{\rightarrow}} \alpha Aw \underset{md}{\rightarrow} \alpha\beta w$ então $A \rightarrow \beta$ na posição seguinte a α é um "handle" de $\alpha\beta w$.

Ex: Seja a gramática

$S \rightarrow aAcBe$

$A \rightarrow A\,b \mid b$

$B \rightarrow d$

e as "formas sentenciais γ = abbcde α = aAbcde

$A \rightarrow b$ na posição 2 é o "handle" de γ

e $A \rightarrow Ab$ na posição 2 é o "handle" de α

Para uma gramática não ambígua toda forma sentencial a direita tem somente um "handle".

O processo de análise sintática por método "shift-reduce" pode ser entendido como uma sucessão de passos de identificação e redução de "handles".

O analisador sintático de "shift-reduce" pode ser implementado com utilização de uma pilha, uma área com a sentença de entrada e o caracter $ utilizado para indicar a base de pilha e o final da sentença de entrada. O analisador sintático pode realizar as seguintes operações:

empilha - o próximo símbolo da sentença de entrada é colocado no topo da pilha.

reduz- o analisador reconhece que o símbolo mais a direita de um "handle" está no topo da pilha, este deve então descobrir o símbolo mais a es -

querda do "handle" e também a que símbolo não terminal reduzir o "handle" considerado.

aceita- o analisador reconhece o final com sucesso da análise sintática.

erro- o analisador descobre um erro e passa o controe à rotina de recuperação de erro.

O analisador começa na situação

Pilha	Entrada	
$	w$	onde w-sentença de entrada

a partir realiza operações de empilha e aceita até encontrar um erro ou até encontrar a situação

Pilha	Entrada
$S	$

que corresponde a operação aceita.

Ex:- Seja a gramática do exemplo anterior e a sentença abbcde, são mostradas abaixo; as operações realizadas, pilha e a área de entrada.

Pilha	Entrada	Ação
$	abbcde	empilha
$a	bbcde	empilha
$ab	bcde	reduz por A → b
$aA	bcde	empilha
$aAb	cde	reduz por A → Ab
$aA	cde	empilha
$aAc	de	empilha
$aAcd	e	reduz por B → d
$aAcB	e	empilha
$aAcBe	$	reduz por S → aAcBe
$S	$	aceita

As diferenças entre os diversos métodos de análise sintática estão na forma em que estes descobrem um "handle" e como conseguem reduzí-lo ao não terminal esquerdo da respectiva produção. Um método para gramáticas de precedência, estava entre as primeiras técnicas utilizadas para análise sintática de linguagens de programação, e existe um bom número de variações do método que se baseia na definição de relação de precedência $\cdot>$ entre símbolos da gramática tal que, percorrendo a forma sentencial $\alpha\beta w$ onde β é o "handle" a relação $\cdot>$ existe entre o último símbolo de β e o primeiro símbolo de w. Esta relação indica uma decisão de redução , se a relação $\cdot>$ não se aplica uma operação empilha deve ser chamada. A localização do lado direito do "handle" é determinado pela relação $\cdot>$, e a determinação do lado esquerdo e que produção reduzir depende do tipo de precedência utilizada.

Os métodos de precedência mais conhecidos são:

- Precedência simples
- Precedência estendida
- Precedência fraca
- Precedência de estrategia mista
- Precedência de operadores

Outros métodos de análise sintática "bottom-up" :

- BRC ("Bounded Right Context")
- Floyd-Evans
- LR

O método LR será detalhado adiante,para detalhes do método BRC e Floyd-Evans ver (AHO ULLMAN[4]).

4.2- Métodos "top-down"

Existem dois métodos mais conhecidos de análise sintática "top-down" .

4.2.1- Método das descidas recursivas

Este método monta a árvore sem necessidade de volta ("backtrack"), e é implementado com a construção de uma subrotina para cada não terminal. Esta subrotina decide que produção do não terminal deve utilizar, analisando o símbolo corrente na entrada e comparando com o primeiro símbolo das produções.

Ex:- Seja a gramática

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | id

```
procedure   E ( );

begin
    T ( );
    ELINHA ( )
end;

procedure   ELINHA ( );

if símbolo entrada = '+'
    then
     begin
      AVANCE ( );
      T( );
      ELINHA( )
end;
```

```
procedure T( )
begin
 F( );
 TLINHA( );
end;

procedure TLINHA( );
if símbolo entrada = '*'
    then
     begin
      AVANCE( );
      F( );
      TLINHA
end;

procedure F( );
if símbolo entrada ='id'
    then AVANCE( )
    else
     if símbolo entrada='('
      then
       begin
       AVANCE( );
       E( );
       if símbolo entrada=')'
        then AVANCE( );
        else ERRO( )
       end
      else ERRO( )
```

4.2.2- Método de análise preditiva

Trabalha com uma tabela previamente calculada, um pilha, a sentença de entrada e umasaída.

A tabela é um matriz referenciada por M(A,a) onde A é um não terminal e a o símbolo corrente na entrada.

O analisador após consulta ao símbolo X do topo da pilha pode realizar as seguintes ações:

1. Se X=a=$ o analisador termina com sucesso a análise sintática da sentença de entrada.

2. Se X=a≠$ o analisador retira X do topo da pilha e passa ao próximo símbolo na entrada

3. Se X é não terminal, o analisador consulta M (X,a). Se a entrada está preenchida coloca-se no lugar de X no topo da pilha, o lado direito da produção preenchida completando a pilha da direita para a esquerda.

   Ex: M (X,a) = X → CB

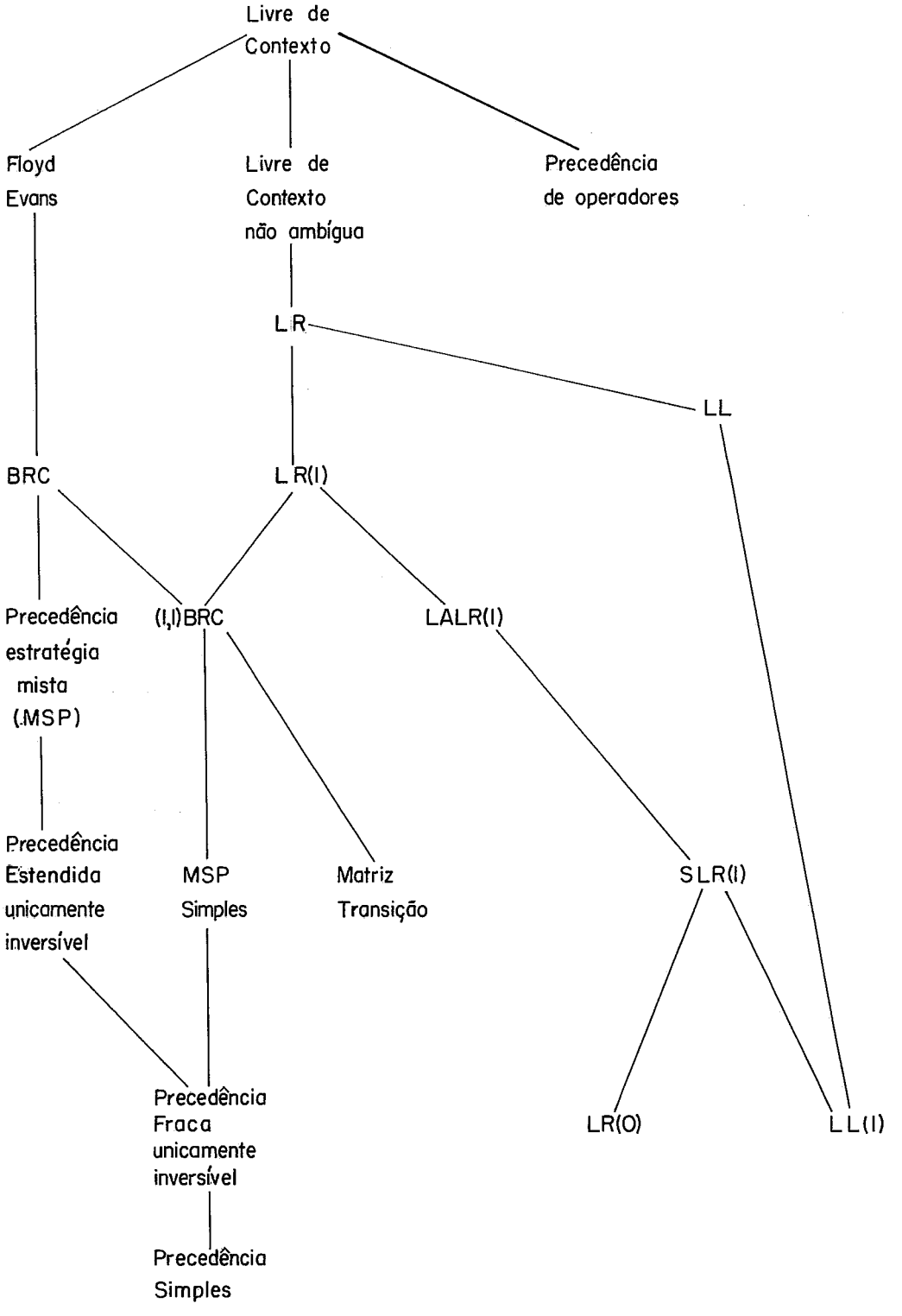
Pilha	Entrada
$AX	aw$
$ABC	aw$

   Se o algorítmo que calcula a tabela deste analisador sintático não contiver entradas múltiplas para o mesmo M (X,a) a gramática é dita LL(1).

4.2.3- Características comuns

Os dois métodos "top-down" acima precisam de ter a gramática preparada através de eliminação de recursividade à esquerda e fatoração a esquerda para que estes funcionem.

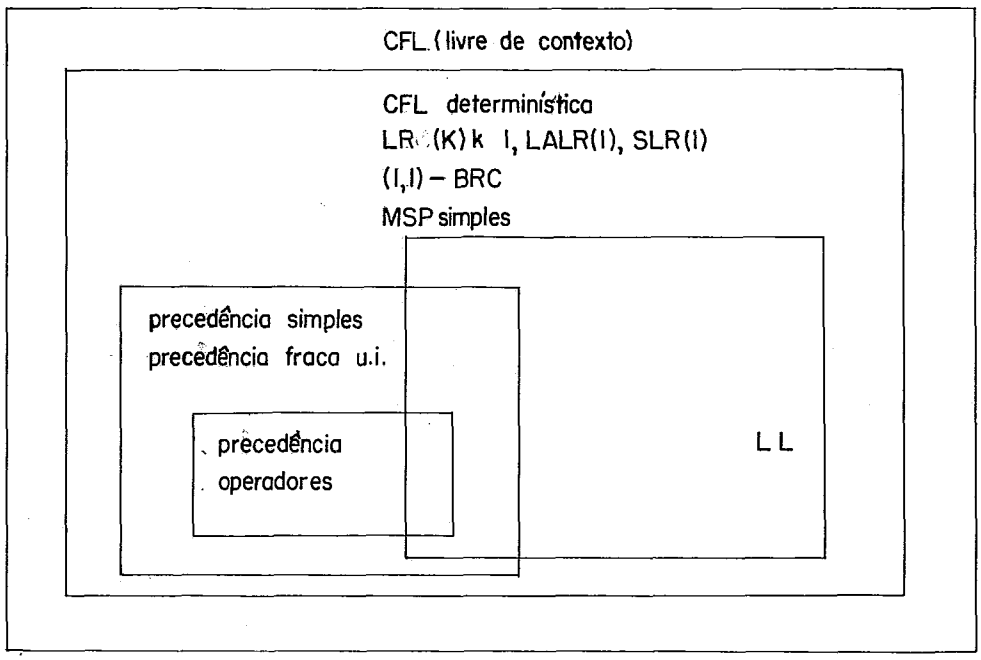
5. Comparação entre classes de gramáticas mais conhecidas :



A figura anterior mostra que existe um analisador sintático LR(1) para a maior parte das gramáticas para métodos de análise sintática conhecidos.

A diferença entre métodos LR(k) para k>1 e LR(1) em relação as gramáticas aceitas é pequena e também entre o método LR(1) para o LALR(1) e SLR(1).Entretanto,o método LR (0) e LL(1) e técnicas de precedência aceitam classes de gramáticas bem menores e necessitam de muitas transforma - ções em suas produções.

6. Comparação entre classes de linguagens mais conheci - das



O fato de que uma técnica de análise sintática rejeita uma certa gramática não significa que rejeitará todas as outras gramáticas desta linguagem. As transformações de uma gramática em outra tem por objetivo convertê-la de forma a aceitar a linguagem original.

Uma linguagem pertence a uma classe se existe pelo menos um reconhecedor que caracteriza esta classe capaz de

aceitá-la. A classe mais extensa conhecida é a classe das linguagens determinísticas que é a mesma das gramáticas LR (k) para k> 1, LALR(1), SLR(1).

É possível transformar qualquer gramática LR(k) em uma gramática SLR(1).

As gramáticas LL estão próprіamente contidas nas linguagens determinísticas. Existem linguagens SLR(1) que não possuem gramática LL(k) o que algumas vezes é utilizado para provar a superioridade das gramáticas LR, porém as linguagens de programação não parecem cair nesta divisão entre linguagens LL e linguagens determinísticas.As inclusões entre classes de linguagens conhecidas é mostrada na figura anterior.

7. Análise sintática LR

7.1- Gramática aumentada derivada de G

Definição:
Seja G=(N,Σ,P,S) uma gramática livre de contexto então

G' - gramática aumentada derivada de G é definida como G={NU{S'},Σ,PU{S'→ S}, S'}. G' é simplesmente G como uma nova produção S'→ S e S' é o novo símbolo inicial não pertencente a N. A nova produção S'→S é numerada com o número 0 e as demais produções de G 1,2,... ....p.

7.2- Gramática LR(k)

Seja G=(N,Σ,P,S) e G'=(N',Σ,P',S') diz-se que

G é LR(k), k> =0 se para que ocorram as três condições

1. $S' \overset{*}{\underset{G'md}{\rightarrow}} \alpha Aw \rightarrow \alpha\beta w$

2. $S' \overset{*}{\underset{G'md}{\rightarrow}} \gamma Bx \underset{G'md}{\rightarrow} \alpha\beta y$

3. $FIRST_k(w) = FIRST_k(y)$

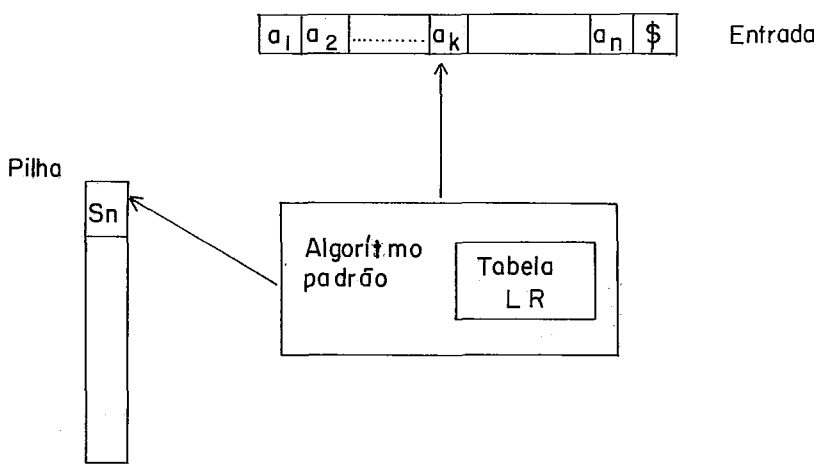
então $\alpha Ay = \gamma Bx$ isto é $\alpha = \gamma$, A=B e x=y.

Uma gramática é LR se é LR(k) para algum k.

7.3- Analisador sintático LR

O analisador sintático LR é um método "bottom - up" do tipo "shift-reduce" onde as tabelas do analisador são utilizadas para indicar as ações de empi - lha, reduz, aceita e erro.

A figura abaixo indica o seu funcionamento.



A entrada é lida da esquerda para a direita e um símbolo de cada vez. A pilha contém a cadeia da forma

$S_0X_1S_1X_2S_2$----------X_mS_m

X_i- símbolo da gramática

S_i- estado qualquer

S_m- estado no topo da pilha

Seja a configuração de um analisador LR o par ordenado tendo como primeiro componente o conteúdo da pilha e o segundo a entrada não expandida.

O analisador determina o estado S_m no topo da pilha , o símbolo a_k constante, e então da configuração $(S_oX_1S_1X_2S_2\cdots\ X_mS_m, a_k\ a_{k+1}\cdots\cdots\ a_n\$)$ pode passar a outras configurações dependendo da consulta as tabelas AÇÃO e GOTO.

A entrada ACAO (S_m, a_k) pode indicar um dos seguintes valores

1. e_i (empilha o estado i)

O analisador empilha o estado S_i e passa a configuração

$$(S_oX_1S_1X_2S_2\cdots\cdots X_mS_m a_k S_i,\ a_{k+1}\cdots a_n\$)$$

onde S_i = GOTO (S_m, a_k) e a_{k+1} passa a ser o símbolo corrente da entrada.

2. r_j (reduzir pela produção j)

O analisador utiliza a produção de número j $(A\rightarrow\beta)$ de tamanho r e reduz r estados e r símbolos da gramática na pilha e passa a seguinte configuração

$$(S_oX_1S_1X_2\cdots X_{m-r}\ S_{m-r}AS,\ a_k a_{k+1}\ldots a_n\$)$$

onde S = GOTO (S_{m-r}, A) e a_k continua como símbolo corrente de entrada.

Os r símbolos da gramática reduzidos são exatamente os mesmos do lado direito da produção j, isto é , iguais a β .

3. acc (aceita)

O analisador termina a análise sintática com sucesso.

4. branco (erro)

As entradas em branco indicam a existência de um erro na construção da sentença de entrada, e deve ser chamada rotina de erro.

Exemplo:

Seja a gramática aumentada

Nº produção	Produção
0	S'→S
1	S →L=R
2	S →R
3	L →*R
4	L →id
5	R →L

A tabela LALR(1) correspondente é

Estado	AÇÃO =	*	id	$	GOTO S	L	R
0		e4	e5		1	2	3
1				acc			
2	e6			r5			
3				r2			
4		e4	e5			8	7
5	r4			r4			
6		e4	e5			8	9
7	r3			r3			
8	r3			r3			
9				r1			

Seja a sentença **id=id. Os movimentos da pilha e sentença de entrada são mostrados a seguir.

Pilha	Entrada	Ação
0	**id=id$	e4
0*4	*id=id$	e4
0*4*4	id=id$	e5
0*4*4 id5	=id$	r4
0*4*4 L8	=id$	r5
0*4*4R7	=id$	r3
0*4L8	=id$	r5
0*4R7	=id$	r3
0L2	=id$	e6
0L2=6	id$	e5
0L2= 6id5	$	r4
0L2= 6L8	$	r5
0L2= 6R9	$	r1
0S1	$	acc

7.4- Construção das tabelas de análise sintática SLR(1)

7.4.1- Item LR(0)

Um item LR(0) de uma gramática G é uma produção de G com um ponto em alguma posição do lado direito.

Exemplo : Seja G

$$
\begin{aligned}
E &\rightarrow TE' \\
E' &\rightarrow +TE' \mid \varepsilon \\
T &\rightarrow FT \\
T' &\rightarrow *FT' \\
F &\rightarrow (E) \mid id
\end{aligned}
$$

São itens LR(0) da produção $E \rightarrow TE'$

$E \rightarrow \cdot TE'$

$E \rightarrow T \cdot E'$

$E \rightarrow TE' \cdot$

A produção $E' \rightarrow \varepsilon$ só tem um item LR(0) representado por $E' \rightarrow \cdot$

Os itens LR(0) são representados no computador por um par de inteiros, o primeiro representando o número da produção e o segundo a posição do ponto dentro da produção.

7.4.2- FECHAMENTO ("CLOSURE") de um conjunto de itens LR(0)

O FECHAMENTO de um conjunto de itens I chamado de CLOSURE(I) é um outro conjunto criado da seguinte forma:

1. Todo item do conjunto pertence ao FECHAMENTO

2. Se $A \rightarrow \alpha \cdot B\beta$ pertence ao FECHAMENTO então os itens $B \rightarrow \cdot\gamma$ são adicionados ao FECHAMENTO se já não estiverem lá.

Exemplo: seja a gramática G do exemplo anterior.
Seja o conjunto de itens de um único item
$I = \{(E \rightarrow T \cdot E')\}$

então CLOSURE (I) contém os seguintes itens

$E \rightarrow T \cdot E'$
$E \rightarrow \cdot + TE'$
$E \rightarrow \cdot$

Algorítmo para cálculo de CLOSURE(I)

```
procedure CLOSURE (I);
begin
  repeat
    for cada item A → α ·Bβ em I e cada produção
        B → γ em G tais que B → ·γ não pertence a I
        do adicione B → ·γ a I
  until não existir itens a adicionar em I;
  return I;
end
```

7.4.3- Definição da função GOTO(I,X)

Seja I um conjunto de itens LR(0) e X um símbolo da gramática.

GOTO(I,X) é o FECHAMENTO(CLOSURE(J)) onde J é o conjunto de itens LR(0) com itens da forma $(A \rightarrow \alpha\ X \cdot \beta)$ tais que $(A \rightarrow \alpha \cdot XI)$ pertençam a I.

Ex: Seja a gramática G

E → TE'
E' → +TE' | ε
T → FT'
T' → *FT'
F → (E) | id

e I_0 conjunto de itens LR(0)

E → ·TE'
T → ·FT'
F → ·(E)
F → ·id

este conjunto de itens possui resultado para a função GOTO para os símbolos da gramática T, F, (e id, calculados abaixo:

GOTO(I,T) : J = {(E → T ·E')}

GOTO(I,T) = CLOSURE (J) : E→ T· E'
E'→·+ TE'
E'→·

GOTO(I,F) : J = { (T → F ·T')}

GOTO(I,F) = CLOSURE (J) : T →F ·T'
T'→·* FT'

GOTO(I,( ) : J = {(F→(·E))}

GOTO(I, ( ) =CLOSURE(J) : F →(·E)

GOTO (I,id) : J = {(F → id •) }

GOTO (I,id) = CLOSURE(J): F→ id•

7.4.4- Construção dos conjuntos de itens LR(0)

A construção das tabelas de análise sintática SLR(1) parte dos conjuntos de itens LR(0), e cada um destes apresenta um estado na análise sintática.

O algorítmo para cálculo dos conjuntos de itens LR(0) é indicado abaixo

```
procedure ITENS(G');
begin
   C: = {CLOSURE ({S' →• S})}
   repeat
      for cada conjunto de itens em C e cada símbolo
          X tal que GOTO(I,X) não é vazio e não exis-
          te em C.
          do adicione GOTO(I,X) a C
    until não existir conjuntos de itens a adicionar
          a C.
```

7.4.5- Cálculo das relações FIRST e FOLLOW em uma gramática

7.4.5.1- FIRST

Para calcular FIRST(X) para todos os símbolos X de uma gramática, basta aplicar as tres regras abaixo até não ser possível adi - cionar mais terminais ou ε a qualquer conjunto FIRST(X):

1. Se X é terminal então FIRST(X) é { X }
2. Se X não é terminal e X → aα é uma produção, então adicione a ao FIRST(X). Se X→ε é uma produção então adicione ε ao FIRST (X).

3. Se $X \rightarrow Y_1Y_2.....Y_k$ é uma produção, então para todo i tal que $Y_1,.....,Y_{i-1}$ sejam não terminais e FIRST(Y_j) contenha ε para $j = 1,2,....i-1$ (i.e $Y_1Y_2....Y_{i-1} \overset{*}{\rightarrow} \varepsilon$), adicione todo símbolo em FIRST(Y_i), mas não ε, ao FIRST(X). Se ε pertence ao FIRST (Yj) para todo j=1,2....,k então adicione ε ao FIRST (X).

Para calcular FIRST de qualquer cadeia $X_1X_2.. ....X_n$ deve-se adicionar ao FIRST($X_1X_2...X_n$) para i = 2,..,n o FIRST(Xi) não considerando o ε se o FIRST(X_{i-1}) contiver ε. Finalmente adiciona-se o ε se FIRST (Xi) contém ε para i=1,. ....,n.

A relação FIRST foi aqui apresentada por ser necessária para o cálculo da relação FOLLOW que por sua vez é utilizada no cálculo das tabelas do analisador sintático SLR(1). A relação FIRST é utilizada para cálculo das tabelas LALR(1) e LR(1) posteriormente apresentadas.

7.4.5.2- FOLLOW

Para calcular FOLLOW(A) para todos os não terminais de uma gramática, basta aplicar as três regras abaixo até não ser possível adicionar terminais a qualquer conjunto FOLLOW(A):

1. $ pertence ao FOLLOW(S) onde S é o símbolo inicial
2. Se existe uma produção $A \rightarrow \alpha B\beta, \beta \neq \varepsilon$ então todo terminal pertencente ao FIRST(β) pertence ao FOLLOW (B).
3. Se existe uma produção $A \rightarrow \alpha B$, ou uma produção $A \rightarrow \alpha B\beta$ onde FIRST(β) contém ε (i.e $\beta \overset{*}{\rightarrow} \varepsilon$), então os terminais pertencentes ao FOLLOW(A) pertencem ao FOLLOW(B).

7.4.6- Cálculo da tabela SLR(1)

Seja $C=\{I_0, I_1, \ldots\ldots, I_n\}$ o conjunto de todos os conjuntos de itens LR(0). As ações do analisador sintático para o estado i correspondente ao conjunto de itens I_i são determinados como a seguir:

1. Se $(A \rightarrow \alpha \cdot a\beta) \in I_i$ e GOTO $(I_i, a) = I_j$ então AÇÃO (i,a) = empilha j (e_j), a é um terminal

2. Se $(A \rightarrow \alpha \cdot) \in I_i$, então AÇÃO (i,a) = reduza por $A \rightarrow \alpha$ para todo a em FOLLOW(A).

3. Se $(S' \rightarrow S \cdot) \in I_i$ então AÇÃO (i,$) = aceita

Se não existem ações conflitantes nas regras acima a gramática é dita SLR(1).

Ex: Seja a gramática

S → CC

C → cC | d

calculando os conjuntos de itens LR(0) de forma já explicada teremos:

I_0: S' → ·S
S → ·CC
C → ·cC
C → ·d

I_1: S' → S·

I_2: S → C·C
C → ·cC
C → ·d

I_3: C → c·C
C → ·cC
C → ·d

I_4 : C → d•

I_5 : S → CC•

I_6 : C → cC•

Temos também para relações GOTO, FIRST, FOLLOW:

GOTO(0,S) = 1, GOTO(0,C) = 2, GOTO(0,c) = 3,
GOTO(0,d) = 4, GOTO(2,C) = 5, GOTO(2,c) = 3,
GOTO(2,d) = 4, GOTO(3,C) = 6, GOTO(3,c) = 3,
GOTO(3,d) = 4

FIRST(S) = {c,d}
FIRST(C) = {c,d}
FOLLOW(S') = {$}
FOLLOW(S) = {$}
FOLLOW(C) = {c,d,$}

e finalmente para a tabela SLR(1) teremos:

	AÇÃO			GOTO	
Estado	c	d	$	S	C
0	e3	e4		1	2
1			acc		
2	e3	e4			5
3	e3	e4			6
4	r3	r3	r3		
5	r1				
6	r2	r2	r2		

7.5- Construção das tabelas de análise sintática LR(1)

7.5.1- Item LR(1)

Um Item LR(1) de uma gramática G é uma produção com um ponto em alguma posição do lado direito e um símbolo terminal chamado "lookahead" do item.

Um prefixo viável é o prefixo da forma sentencial a direita que não contém símbolos a direita do "handle". Um item LR(1) $(A \rightarrow \alpha \cdot \beta, a)$ é válido para um prefixo viável γ se existe a derivação $S \overset{*}{\underset{rm}{\rightarrow}} \delta Aw \rightarrow \delta\alpha\beta w$, onde $\gamma = \delta\alpha$ e a é o primeiro símbolo de w ou w é ε e a é \$.

Ex: Seja a gramática

$S \rightarrow CC$

$C \rightarrow cC \mid d$

Existe uma derivação a direita $S \overset{*}{\underset{rm}{\rightarrow}} ccCcd \overset{\rightarrow}{\underset{rm}{}}$

cccCcd o item $(C \rightarrow c \cdot C, c)$ é válido para o prefixo viável $\gamma = ccc$ pois $\delta = cc$, $A = C$, $w = cd$, $\alpha = c$ e $\beta = C$ na definição dada acima.

$(C \rightarrow c \cdot C, \$)$ é válido para o prefixo Ccc pois

$S \overset{*}{\underset{rm}{\rightarrow}} CcC \overset{\rightarrow}{\underset{rm}{}} CccC$ e $\delta = Cc$ $A = C$ $w = \varepsilon$ $\alpha = c$

$\beta = C$

Os itens LR(1) são representados no computador por um terno de inteiros o primeiro representando o número da produção, o segundo a posição do ponto dentro da produção, e o terceiro o número do terminal "loakahead".

7.5.2- FECHAMENTO ("CLOSURE") de um conjunto de itens LR(1)

O FECHAMENTO de um conjunto de itens I chamado CLOSURE(I) é um outro conjunto de itens criado da seguinte forma:

1. Todo item do conjunto pertence ao FECHAMENTO
2. Se $(A \rightarrow \alpha \cdot B\beta, a)$ pertence ao FECHAMENTO, então os

itens (B → •γ ,b) onde b pertence ao FIRST (βa) são adicionados ao FECHAMENTO se já não estiverem lá.

Seja a gramática G

S' → E
E → TE'
E' → +TE'|ε
T → FT'
T' → *FT'
F → (E)| id

e seja I = { (S'→ •E , $)}

a relação FIRST tem os seguintes valores

FIRST(E) = (, id
FIRST(T) = (, id
FIRST(E')= +, ε
FIRST(F) = (, id
FIRST(T')= *

aplicando as regras para cálculo de CLOSURE (I) obtém-se

(S' →•E, $)
(E →•TE',$)
(T →•FT',+)
(T →•FT',$)
(F →•(E),*)
(F →•id,*)

Algorítmo para cálculo de CLOSURE(I)

```
procedure CLOSURE(I)
begin
    repeat
        for cada item (A →α•Bβ,a) em I, cada
            produção B →γ , e cada terminal b
            em FIRST (βa) tal que (B→• γ ,b)
            não pertence a I
            do adicionar (B →•γ, b) a I;
    until não existir itens a adicionar a I.
    return I
end;
```

7.5.3- Definição da função GOTO(I,X)

Seja I um conjunto de itens LR(1) e X um símbolo da gramática.

GOTO(I,X) é o FECHAMENTO (CLOSURE(J)) onde J é o conjunto de itens LR(1) com itens da forma (A→ α X •β,a) tais que (A →α•Xβ ,a)pertençam a I.

Ex:- Seja a mesma gramática G e conjunto de itens LR(1) do exemplo anterior

```
I : S' →•E, $
    E  →•TE', $
    T  →•FT', +
    T  →•FT', $
    F  →•(E), *
    F  →•id, *
```

este conjunto de itens possui resultado para a função GOTO para os símbolos da gramática E,T, F,(e id calculados abaixo :

GOTO(I,E) : J= {(S' → E•,$)}

GOTO(I,E)=CLOSURE(J): S' → E• ,$

GOTO(I,T) : J= {(E → T•E', $)}

GOTO(I,T)=CLOSURE(J): E → T•E', $
E'→ •+TE', $
E → •, $

GOTO(I,F) : J= {(T→ F• T', +) , (T →F •T', $)}

GOTO(I,F)=CLOSURE(J): T→ F•T', +
T→ F•T', $
T→ •* FT', +
T→ •* FT', $

GOTO(I,() = J={(F →(•E), *)}

GOTO(I,F)=CLOSURE(J): F → (•E), *
E →•TE',)
T →•FT',+
T →•FT',)
F →•(E), *
F →•id, *

GOTO(I,id) : J = { (F → id •, *)}

GOTO(I,id)= CLOSURE(J)=F → id•,*

7.5.4- Construção dos conjuntos de itens LR(1)

A construção das tabelas de análise sintática LR(1) parte dos conjuntos de itens LR(1), e cada um destes representa um estado da análise sintática.

O algorítmo para cálculo dos conjuntos de itens LR(1) é indicado a seguir.

```
procedure GOTO(I,X);

begin
    Seja J o conjunto de itens (A→ αX•ß,a) ,
    tal que (A→α•Xβ, a) pertence a I;
    return CLOSURE(J)
end;


procedure ITENS(G')
begin
    C:= {CLOSURE ({ S'→  •S,$} ) }
    repeat
      for cada conjunto de itens I em C e ca-
          da símbolo da gramática X tal  que
          GOTO(I,X) não é vazio e não perten-
          ce a C
          do adicionar GOTO(I,X) a C
     until não existir conjuntos de itens a a
           dicionar a C
end
```

7.5.5- Cálculo da tabela LR(1)

Seja C = $(I_i, I_1, \ldots, I_n)$ o conjunto de todos os itens LR(1)As ações do analisador sintático para o estado i correspondente ao conjunto de itens I são determinadas como a seguir:

1. Se $(A \to \alpha \cdot a\beta\ ,\ b) \in I_i$ e $GOTO(I_i, a) = I_j$ então AÇÃO (i,a) = empilha j (e_j), a é um terminal.

2. Se $(A \to \alpha \cdot, a) \in I_i$ então AÇÃO (i,a) = reduza por $A \to \alpha$

3. Se $(S' \rightarrow S\cdot, \$) \in I_i$, então AÇÃO (i,$)= aceita

Se não existem ações conflitantes nas regras acima a gramática é dita LR(1).

Seja a gramática G

S' → S
S → CC
C → cC | d

calculando os conjuntos de itens LR(1) obtem-se:

I_o: S' →•S,$
S →•CC,$
C →•cC, c|d
C →•d , c d

I_1: S' → S •,$

I_2: S → C •C, $
C → •cC, $
C → •d, $

I_3 :C → c •C, c|d
C → •cC , c|d
C → •d , c|d

I_4: C → d•, c|d

I_5: S → CC •, $

I_6: C → c•C, $
C → •cC, $
C → •d, $

I_7: C → d•, $

I_8: C → cC•, c|d

I_9: C → cC•, $

da função GOTO tem-se

GOTO(0,S)=1, GOTO(0,C)=2, GOTO(0,c)= 3, GOTO(0,d) = 4
GOTO(2,C)=5, GOTO(2,c)=6, GOTO(2,d)= 7
GOTO(3,C)=8, GOTO(3,c)=3, GOTO(3,d)= 4
GOTO(6,C)=9, GOTO(6,c)=6, GOTO(6,d)= 7

e finalmente a tabela LR(1):

	AÇÃO			GOTO	
Estado	c	d	$	S	C
0	e3	e4		1	2
1			acc		
2	e6	e7			5
3	e3	e4			8
4	r3	r3			
5			r1		
6	e6	e7			9
7			r3		
8	r2	r2			
9			r2		

7.6- Construção das tabelas de análise sintática LALR(1)

7.6.1- Método simplificado

Realizar os seguintes passos:

1. Construir $C = \{I_0, I_1, \ldots, I_n\}$, a coleção dos conjuntos de itens LR(1).

2. Para conjuntos de itens que tenham as mesmas produções e posições do ponto, substituir estes conjuntos pela sua união.

3. Seja $C' = (J_0, J_1, \ldots J_n)$ os conjuntos de itens LR(1) resultantes. As ações do analisador sintático LALR(1) para o estado i são construídos do conjunto de itens J_i de maneira idêntica a da construção para tabelas LR(1). Se não existem conflitos após aplicação daquele algorítmo, a gramática é dita LALR(1).

4. As tabelas da função GOTO são construídas como se segue.
   Se J é a união de um ou mais conjuntos de itens com as mesmas produções e posições de ponto, i.e., $J = I_1 U I_2 U \ldots U I_m$ e K a união dos conjuntos que tem as mesmas produções e posições do ponto de $GOTO(I_1, X)$ então $GOTO(J, X) = K$.

Seja a gramática:

S' → S
S → CC
C → cC | d do exemplo anterior

Dos conjuntos de itens LR(1) do exemplo ante-

rior, verifica-se que podem ser unidos os seguintes conjuntos:

$I_{36} = I_3 UI_6, I_{47} = I_4 UI_7, I_{89} = I_8 UI_9$

As funções GOTO ficam da seguinte forma:

GOTO(0,S) = 1, GOTO(0,C) = 2, GOTO(0,c) = 36
GOTO(0,d) = 47,GOTO(2,C) = 5, GOTO(2,c) = 36
GOTO(2,d) = 47,GOTO(36,C)=89, GOTO(36,c)= 36
GOTO(36,d)= 47

e finalmente para tabelas LALR(1) tem-se

	AÇÃO			GOTO	
Estado	c	d	$	S	C
0	e36	e47		1	2
1			acc		
2	e36	e47			5
3	e36	e47			89
4	r3	r3	r3		
5			r1		
6	r2	r2	r2		

Este método apesar de simples gasta muito espaço pois parte dos conjuntos de itens LR(1). A seguir será apresentado algorítmo, o programado na tese, que parte do conjunto de itens LR(0) com número muito menor que o LR(1).

7.6.2- Método para menor espaço

7.6.2.1- Item central

Os itens centrais são aqueles não adicionados a um conjunto de itens atraves do FECHAMENTO destes.

Um item central é reconhecido facilmente pois ou é o item $(S' \rightarrow \bullet S, \$)$, para item central LR(0) é $(S' \rightarrow \bullet S)$, ou possui um símbolo a esquerda do ponto.

7.6.2.2- "Lookaheads" propagados e gerados espontaneamente

Seja o item $(A \rightarrow \delta \bullet C\gamma , a)$ em I_1, $C \overset{*}{\underset{md}{\rightarrow}} E\alpha$ e $E \rightarrow X\beta$ uma produção, podendo $\alpha = \varepsilon$ e $C = E$ os itens $(E \rightarrow X \bullet \beta , b)$ tais que $b \in$ FIRST $(\alpha\gamma)$ pertencem ao GOTO(I_1, X) e diz-se que b é gerado espontaneamente como "lookahead" para $E \rightarrow X \beta$.

Se na situação acima $\alpha\gamma \overset{*}{\rightarrow} \varepsilon$ então os itens $(E \rightarrow X \bullet \beta , a)$ também pertencem ao GOTO(I_1, X).

Diz-se neste caso que os símbolos a são propagados como "lookaheads" de $A \rightarrow \delta \bullet C\gamma$ para $E \rightarrow X \bullet \beta$.

Um algorítmo que determina quando um item LR(1) em I gera um "lookahead" espontaneamente em GOTO(I,X) ou propaga é indicado a seguir:

Entrada: O conjunto K de itens centrais em um conjunto I de itens LR(0), e um símbolo da gramática X.

Saída: Para cada item central $A \rightarrow \alpha X \bullet \beta$ de GOTO(I,X), é produzido o conjunto de "lookaheads" gerados espontaneamente de itens em I e aqueles itens

centrais em I que propagam "lookaheads" para A→αX•β em GOTO(I,X).

Método: O algorítmo usa um símbolo fictício # para detectar situações em que os "lookaheads" são propagados.

for cada item B →γ•δ em K do

begin

J' := CLOSURE ({(B→γ•δ,#)})

if (A→α•Xβ,a) ε J' onde a não é #

then o "lookahead" a é gerado espontaneamente para item A →αX•β em GOTO(I,X).

if (A→α•Xβ,#) εJ',

then o "lookahead" é propagado de B → γ•δ em I para A→αX•β em GOTO(I,X)

end

7.6.2.3- Construção dos conjuntos de itens LALR(1)

Entrada: A gramática G aumentada pela produção S'→S.

Saída: Os conjuntos de itens LALR(1) de G.

Método:

1. Construir os conjuntos K formado pelos itens centrais dos conjuntos I de itens LR(0) de G.
2. Pelo algorítmo anterior determina-se para cada item LR(0) (A→α•β) em K, e para cada símbolo da gramática X em que itens centrais (B→γ•δ) de GOTO(I,X) os

"lookaheads" são propagados e A→α•β em I para B→ γ•δ em GOTO(I,X).

Determina-se também os "lookaheads" gerados espontaneamente para B→γ•δ em GOTO(I,X).

3. Cria-se uma pilha de ternos (I,A→α•β,a) onde I é um conjunto de itens LR(0), A→α•β um item central de I e a um símbolo terminal tal que (A→α•β,a) pertence ao conjunto de itens LALR(1) que possui as mesmas produções e posições do ponto que I. Utiliza-se um vetor de "bits" para identificar se um terno já foi inserido através da rotina INSERE.

O algorítmo abaixo cria para cada conjunto de itens LR(0) I, e para cada item central (A→α•β) de I, uma lista dos conjuntos de "lookheads" para A→α•β no conjunto de itens LALR(1) cujas produções e posições do ponto são iguais as de I.

Algorítmo:

```
procedure INSERE (I,A →α•β,a);
if not ON (I,A→α•β,a)
  then
   begin
    coloca (I,A→α•β,a) em PILHA;
    ON (I,A→α•β,a):= true;

end
```

```
begin
  for todo I,A→α•β e a do ON(I,A→α•β,a):=false:
  PILHA : = vazio;
  INSERE(I0,S'→•S,$);
  for cada I,A→α•β e a tal que a é gerado esponta-
      neamente como "lookahead" de A→α•β em I
      do INSERE (I,A→α•β,a);

  while PILHA não vazia do
   begin
      retira (J,B→γ•δ,a), o terno do topo da PILHA
      for cada símbolo da gramática X do
          for cada item central A→α•β do GOTO(J,X)
              tal que B→γ•δ propaga "lookaheads" pa
              ra A→α•β em GOTO(J,X) do INSERE(GOTO
              (J,X), A→α•β,a)
   end
  end
```

III- IMPLEMENTAÇÃO

1. Escolha da Linguagem

A linguagem escolhida foi o ALGOL para o computador Burroughs B6700 pelos seguintes motivos:

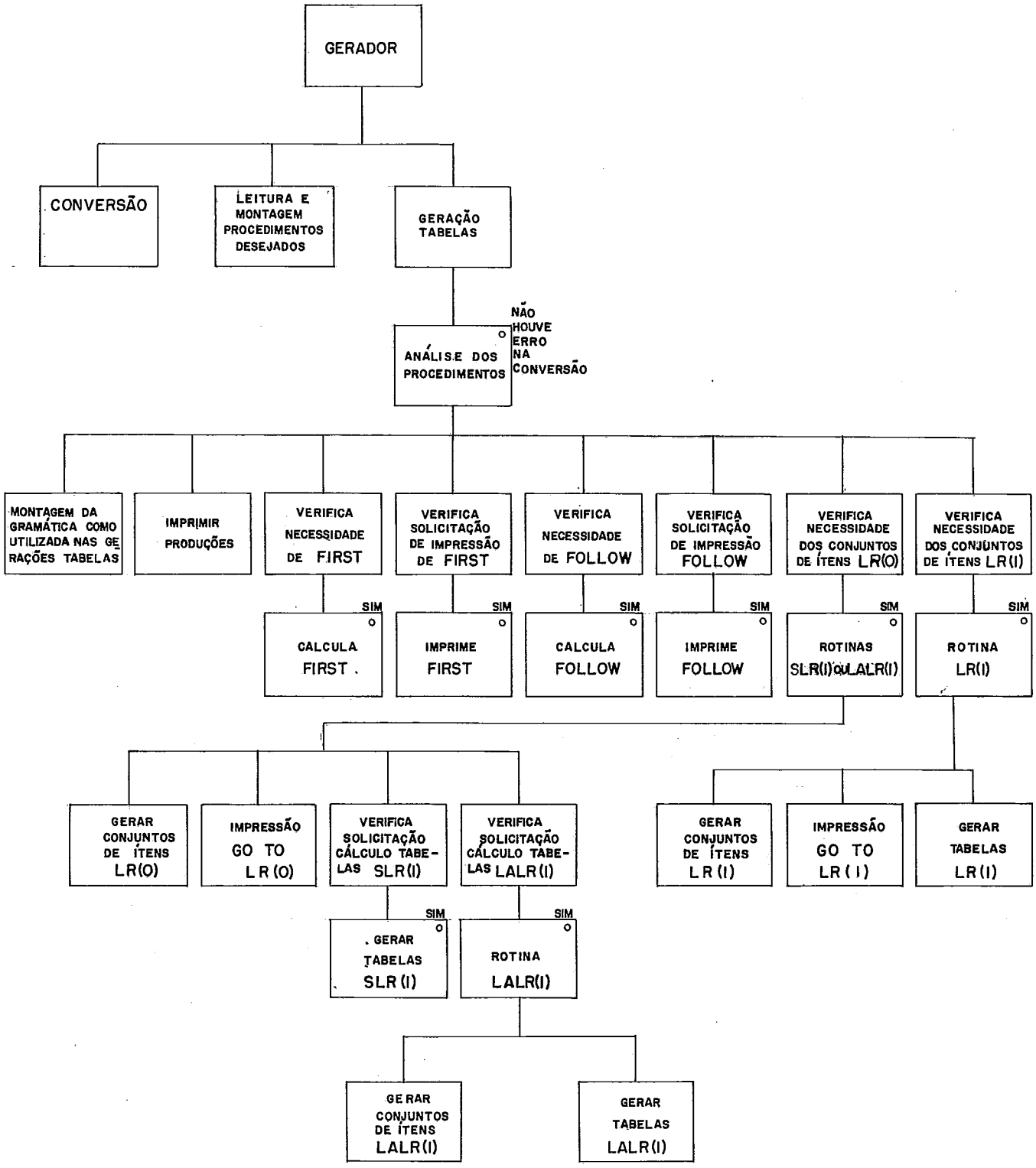
- Este computador está instalado no Núcleo de Computação Eletrônico e é o que atende a maior parte dos usuários deste Núcleo.

- A linguagem ALGOL tem uma estrutura para a programação dos algorítmos que permite fácil compreensão, documentação e manutenção destes.

- Esta linguagem é bastante difundida no meio científico criando possibilidade de utilização do produto desta tese em outros locais.

- Apesar de ser uma linguagem de alto nível esta possui características que permitem implementação eficiente dos algorítmos.

2. Descrição Geral do Gerador



Inicialmente uma etapa de conversão é realizada. Esta etapa consiste em a partir dos cartões contendo a gramática de entrada (V.1.1.1), realiza análise sintática e geração de estrutura interna temporária desta gramática. São também impressos os relatórios; gramática de entrada e mensagens de erro, representação interna de terminais e representação interna dos não terminais.

A seguir obtém-se da leitura do cartão parâmetro PROCED os procedimentos desejados. Ver (V.1.1.2) para maiores detalhes.

A geração das tabelas é então iniciada e só é realizada se não houve erro grave na codificação da gramática de entrada . Esta geração consiste dos seguintes passos:

i- Passagem da estrutura interna temporária da gramática montada pela conversão para estrutura utilizada pelos algorítmos .

ii- Impressão das produções com os símbolos na sua representação interna.

iii- Cálculo da relação FIRST para a gramática de entrada se algum dos procedimentos solicitados precisam deste cálculo.

iv- Impressão de relatório com a relação FIRST para todos os símbolos não terminais da gramática de entrada, se o procedimento FIRST foi solicitado no cartão PROCED.

v- Cálculo da relação FOLLOW para a gramática de entrada se algum dos procedimentos solicitados precisam deste cálculo.

vi- Impressão de relatório com a relação FOLLOW para todos os símbolos não terminais da gramática de entrada se o procedimento FOLLOW foi solicitado no cartão PROCED.

vii- Se existe a necessidade de cálculo dos conjuntos de itens LR(0) o que ocorre se o cartão PROCED requer procedimentos SLR ou LALR1, são realizados na ordem:

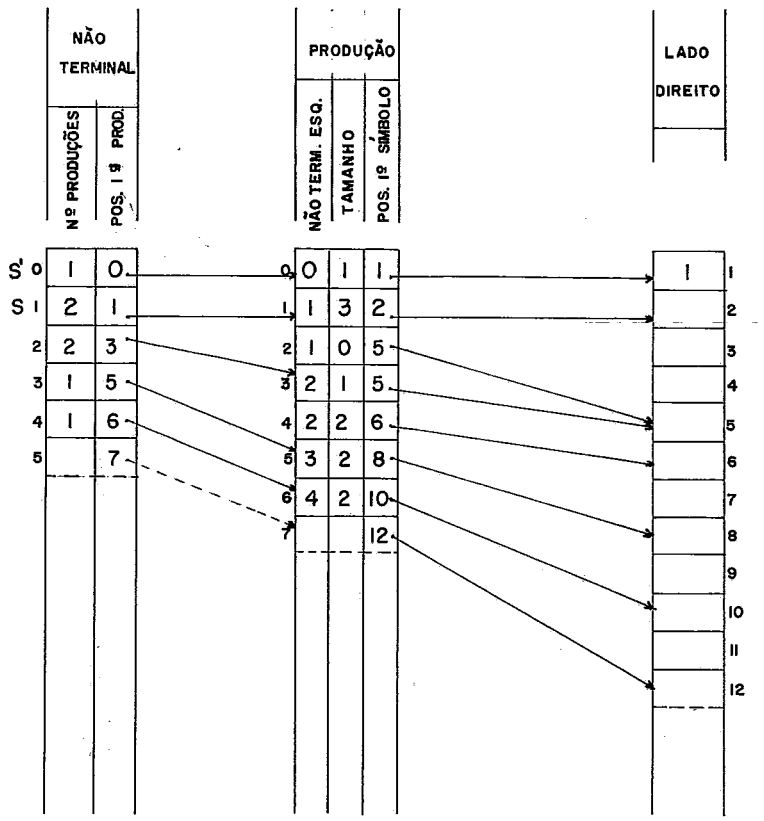
- Cálculo dos conjuntos de itens LR(0)
- Impressão de relatório com a relação GOTO para estes conjuntos de itens.
- Se o cartão PROCED requer o procedimento SLR, calcula-se a tabela para análise sintática SLR(1). São impressos também os relatórios; conjuntos de itens LR(0), tabela de análise sintática SLR(1) e conflitos na tabela de análise sintática SLR(1).
- Se o cartão PROCED requer o procedimento LALR1, geram-se os conjuntos de itens LALR1 com algorítmo que parte dos conjuntos de itens LR(0), e a seguir calcula-se a tabela para análise sintática LALR(1). São impressos também os relatórios; conjuntos de itens LALR(1), tabela de análise sintática LALR(1) e conflitos na tabela de análise sintática LALR(1).

viii-Se existe a necessidade de cálculo dos conjuntos de itens LR(1), o que ocorre se o cartão PROCED requer procedimento LR1, são realizados na ordem:

- Cálculo dos conjuntosde itens LR(1).
- Impressão de relatório com a relação GOTO para estes conjuntos de itens.
- Cálculo das tabelas para análise sintática LR(1).São impressos também os relatórios; conjuntos de itens LR(1), tabela de análise sintática LR(1) e conflitos na tabela de análise sintática LR(1).

3. Representação das estruturas importantes utilizadas

3.1- Representação da gramática de entrada



A gramática é representada por 3 vetores.

O primeiro vetor contém características dos não terminais da gramática. O índice do vetor corresponde a representação interna utilizada para este não terminal, observar que o não terminal aumentado S' tem representação interna igual a zero. Cada elemento deste vetor contém o par (nº de produções do não terminal, posição da primeira produção do não terminal).

O segundo vetor contém características das produções. O índice do vetor corresponde a representa -

ção interna da produção, observar que a produção S'→ S tem representação interna igual a zero. Cada elemento deste vetor contém o terno (Não terminal esquerdo da produção, tamanho da produção, posição do primeiro símbolo da produção).

O terceiro vetor contém os símbolos do lado direito de todas as produções. Cada elemento contém a representação interna do símbolo da gramática de entrada.

Cada elemento de qualquer um dos 3 vetores ocupa uma palavra de 48 "bits".

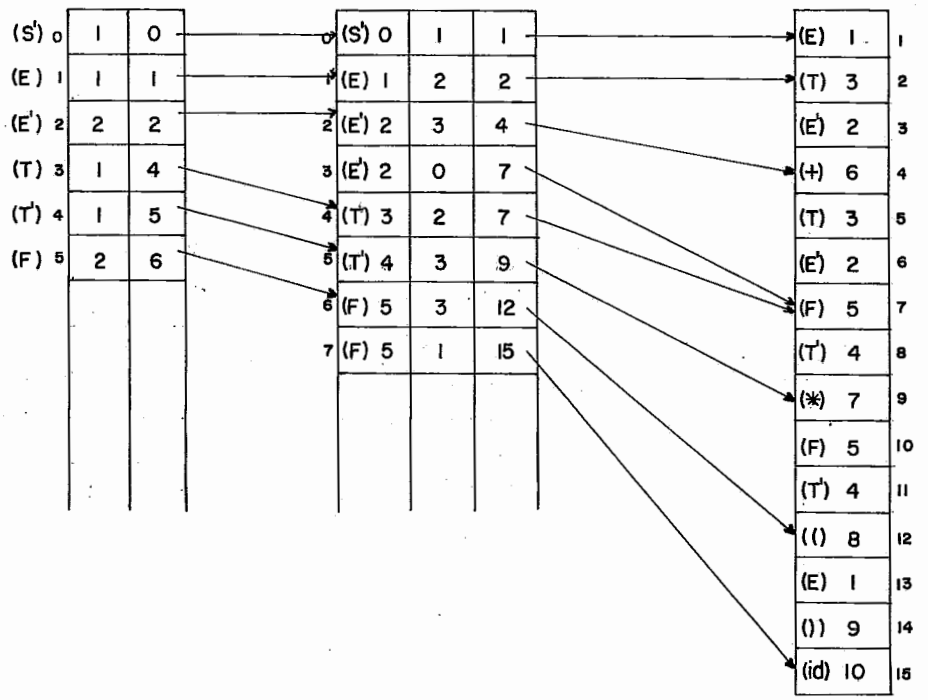
Ex: Seja a gramática

(0) S' → E
(1) E → TE'
(2) E' → +TE'
(3) E' → ε
(4) T → FT'
(5) T' → *FT'
(6) F → (E)
(7) F → id

Considerando as representações internas:

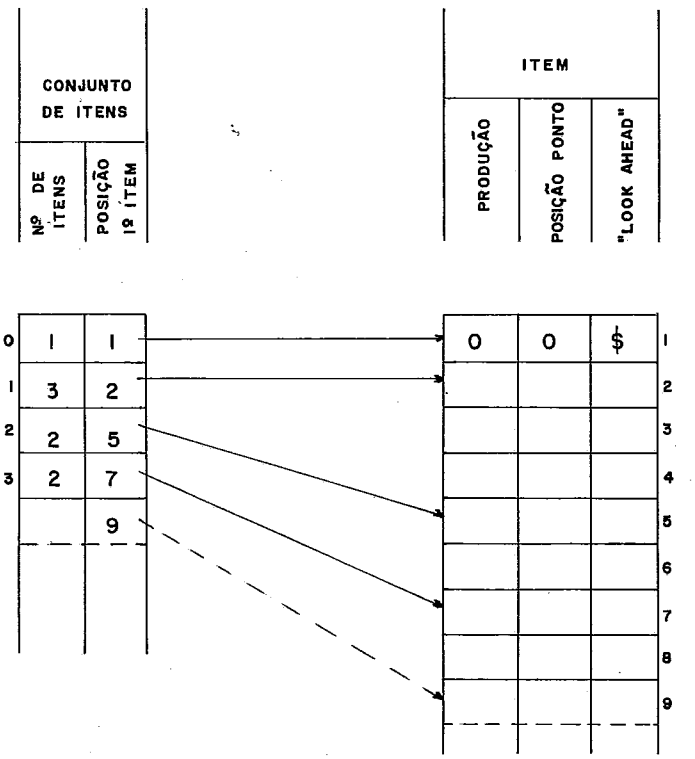
S'	0
E	1
E'	2
T	3
T'	4
F	5
+	6
*	7
(	8
)	9
id	10

A gramática será representada da seguinte forma:



Os símbolos foram colocados para melhor compreensão do desenho.

3.2- Representação dos conjuntos de itens LR(0) e LR(1)



Os conjuntos de itens são representados por 2 vetores.

O primeiro vetor contém características dos conjuntos de itens. O índice do vetor corresponde a representação numérica interna deste conjunto de itens. Cada elemento deste vetor contém o par (nº de itens centrais do conjunto de itens, posição do primeiro item).

O segundo vetor contém características dos itens. Cada elemento contém para um item LR(0) o par (produção, posição do ponto), e para um item LR(1) o terno (produção, posição do ponto, "lookahead").

Cada elemento do primeiro ou segundo vetor ocupa uma palavra de 48 "bits".

Seja a gramática

(o)	S'	→	S	onde	S'	- 0
(1)	S	→	CC		S	- 1
(2)	C	→	cC		C	- 2
(3)	C	→	d		c	- 3
					d	- 4
					$	- 5

Os conjuntos de itens LR(1) considerando itens centrais são os seguintes:

I_o: S' → •S,$

I_1: S' → S•, $

I_2: S → C•C,$

I_3: C → c•C, d|d

I_4: C → d•, c|d

I_5: S → CC• ,$

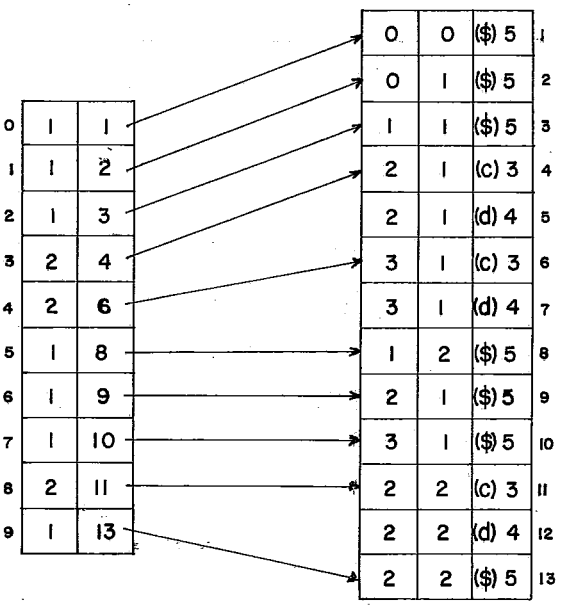
I_6: C → c•C, $

I_7: C → d• ,$

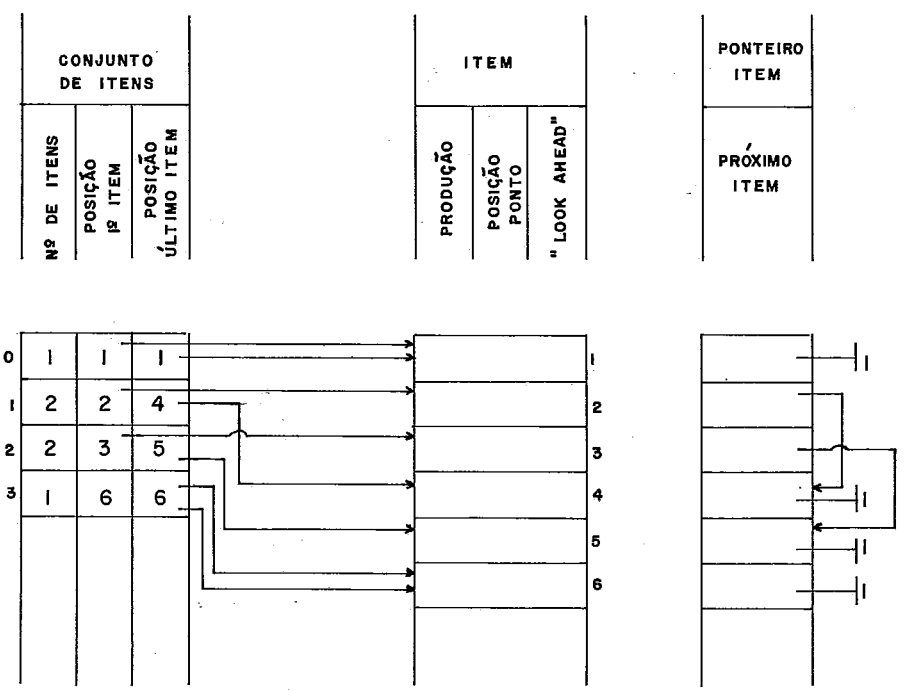
I_8: C → cC• ,c|d

I_9: C → cC• ,$

Estes são representados internamente da seguinte forma:



3.3- Representação dos conjuntos de itens LALR(1)



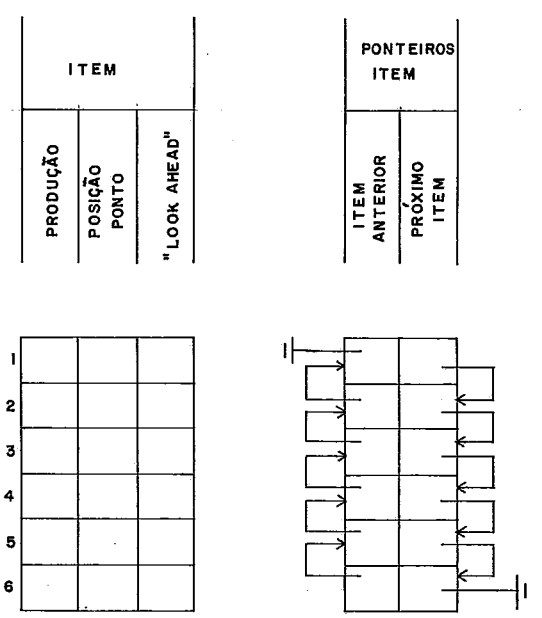
Os conjuntos de itens são representados por 3 vetores.

O primeiro vetor contém características dos conjuntos de itens. O índice do vetor corresponde a representação numérica interna deste conjunto de itens. Cada elemento deste vetor contém o terno (nº de itens centrais do conjunto de itens, posição do primeiro item, posição do último item).

A parte relativa aos itens LALR(1) é formada de uma lista encadeada devido a criação não seqüencial destes itens. O segundo vetor é idêntico aos itens LR(0) e LR(1) anterior e o terceiro vetor contém os ponteiros que mantém a lista.

3.4- Fechamento de um conjunto de itens (CLOSURE)

Os conjuntos de itens são representados apenas pelos seus itens centrais, é portanto necessário nos algorítmos desta tese para geração das tabelas, realizar a operação CLOSURE(I) para obtenção do conjunto de itens completo. A sua representação é indicada a seguir.



O resultado da CLOSURE(I) pode ser usado de forma sequencial utilizando somente a parte relativa ao item, ou como uma lista duplamente encadeada quando há necessidade de retirar elementos deste fechamento ao longo de uma análise. No segundo caso a alocação inicial dos ponteiros é como indicada acima.

Cada item é representado por um terno (produção, posição ponto, "lookahead") e está contido em uma palavra de 48 "bits". Os ponteiros do item também estão contidos em uma palavra de 48 "bits".

3.5- Representação da relação FIRST dos símbolos não terminais da gramática de entrada

Não Terminal	Terminais 1...2 ..... J ..... T...$	ε
1		
2		
3		
⋮		
I		
⋮		
N		

N - nº de não terminais
T - nº de terminais

O terminal de representação interna J pertence ao FIRST do não terminal de representação interna I se o "bit" relativo ao par (I,J-N) estiver ligado.

Para se verificar se o não terminal gera ε utiliza-se um vetor de "bits" a parte.

Considerando-se a utilização de palavras de 48 "bits" para representar as linhas da Matriz, a Matriz que contém os dados acima tem os seguintes índices MATFIRST (1:N,0 : $\lfloor (T-1)/48 \rfloor$), onde $\lfloor x \rfloor$ representa a parte inteira de x.

3.6- Representação da relação FOLLOW dos símbolos não terminais da gramática de entrada

Idêntico a relação FIRST com os seguintes índices MATFOLLOW (0:N,0:$\lfloor (T-1)/48 \rfloor$)

4.- Limitações físicas importantes

4.1- Número máximo de símbolos da gramática

O número máximo de símbolos terminais e não terminais é limitado para efeito de utilização nos algorítmos pelo vetor com características das produções na primeira parte de seu elemento (ver III.3.1). Este valor é o maior inteiro contido em 16 "bits" que é igual a $(2^{16}-1)=65535$.

4.2- Número máximo de produções

Por restrição idêntica à anterior o número máximo de produções é 65535.

4.3- Número máximo de conjuntos de itens

Este número é limitado pela maior representação interna possível para um conjunto de itens que é igual a $10^{6}-1$ ("format" I6 utilizado nos relatórios). Os algorítmos possuem valor máximo igual ao maior inteiro possível na representação INTEGER do ALGOL.

4.4- Limites atribuídos por valor "default"

Os valores de nº de itens para realização do FECHAMENTO de um conjunto de itens; total de conjuntos de itens e total de itens LR(0), LALR(1), LR(1); total de símbolos não terminais, símbolos do lado direito das produções e área para guardar nomes destes símbolos utilizados na conversão da gramática de entrada, são atribuídos por valor "default" se não existir o cartão parâmetro com o valor respectivo (Ver V.1.1).

4.5- Comparação com gramática grande

Os valores máximos relacionados acima não podem ser considerados uma restrição pois uma gramática de tamanho grande tem uma ordem de centenas de produções e símbolos, milhares ou centenas de conjuntos de itens e itens LR(0) e LALR(1) e milhares ou dezena de milhares conjuntos de itens e itens LR(1).

5. Tamanho das áreas utilizadas

Uma palavra do ALGOL tem 48 "bits" ou 6 "bytes". O cálculo será feito em qualquer uma destas unidades. A conversão de uma para outra é elementar.

A alocação dinâmica da linguagem ALGOL é constantemente utilizada, e a partir do instante em que uma área não é mais necessária ela é liberada.

A seguir serão detalhados por bloco no diagrama (III-1) as áreas que são ativadas neste bloco. Observar que se uma área está ativa em um bloco ela também está ativa nos blocos de níveis mais baixos. As áreas mencionadas são apenas aquelas que dependem de parâmetros relativos à gramática de entrada.

As áreas definidas por variáveis de trabalho são desprezíveis nas gramáticas utilizadas em uma linguagem de pro

gramação normal, e se o tamanho destas gramáticas é pequeno o espaço ocupado pelas variáveis de trabalho deixa de ser uma preocupação importante.

Gerador:

- Gramática em representação intermediária após análise pela conversão
  Tamanho-(nº de não terminais + nº de símbolos do lado direito das produções x 2) palavras

Conversão:

- Nome dos não terminais e terminais
  Tamanho- 3 x (nº de caracteres dos nomes dos símbolos + nº total de símbolos) "bytes"

Geração de Tabelas:

- Gramática de entrada na representação (III.3.1)
  Tamanho (Nº de não terminais + Nº de produções + Nº de símbolos do lado direito das produções) palavras

- Área para realização do fechamento de um conjunto de itens qualquer
  Tamanho 2 x (nº de itens do conjunto de itens com o maior nº destes itens) palavras

- Matriz deFIRST dos não terminais da gramática de entrada
  Tamanho ((nº de não terminais-1) x (nº de terminais + 2))"bits"

- Matriz deFOLLOW dos não terminais da gramática de entrada

Tamanho-((nº de não terminais) x (nº de terminais +1))
"bits"

Rotinas SLR(1)/LALR(1):

- Conjunto de itens LR(0) e itens LR(0)
  Tamanho (nº de conjuntos de itens LR(0) + nº de itens centrais LR(0))palavras

- Matriz GOTO para conjuntos LR(0) e LALR(1)
  Tamanho (nº de conjuntos de itens LR(0) x nº de símbolos) palavras

- Matriz para identificar se as produções de um não terminal foram adicionadas ao fechamento de um conjunto de itens LR(0)
  Tamanho- (nº de não terminais)"bits"
  Observação - Esta área por estar definida dentro da rotina que realiza o fechamento LR(0), só é ativada quando esta rotina é chamada.

Geração das tabelas SLR(1):

- Vetor para identificar ações na tabela de análise sintática SLR(1) para um estado qualquer
  Tamanho- (nº de terminais + 1) palavras

Rotina LALR(1):

- Conjunto de itens LALR(1) e itens LALR(1)
  Tamanho- (nº de conjuntos de itens LALR(1) + 2 x (nº de itens centrais LALR(1)) palavras

- Matriz para identificar se as produções de um não terminal com respectivos "lookaheads" foram adicionados ao fechamento de um conjunto de itens LALR(1).

Tamanho- ((Nº de não terminais-1) x (nº de terminais + 1))"bits"

Observação- Esta área por estar definida dentro da rotina que realiza o fechamento LALR(1), só é ativada quando esta rotina é chamada.

Geração dos conjuntos de itens LALR(1):

- Pilha para geração dos conjuntos de itens LALR(1)
  Tamanho- 2 x(nº de itens centrais LALR(1)) palavras

- Matriz para realizar fechamento de um item LALR(1) com terminal fictício # ("dummy lookahead").
  Tamanho- ((nº de não terminais-1) x (nº de terminais + 2))"bits"

  Observação- Esta área por estar definida dentro da rotina que realiza o fechamento LALR (1) com terminal #, só é ativada quando esta rotina é chamada.

- Matriz e lista com itens LR(0) propagados
  Tamanho- ((Nº de produções x Tamanho da maior produção) + nº de itens centrais LALR(1))

Geração das tabelas LALR(1):

- Vetor para identificar ações na tabela de análise sintática LALR(1) para um estado qualquer
  Tamanho - (nº de terminais + 1) palavras

Rotina LR(1):

- Conjunto de itens LR(1) e itens LR(1)
  Tamanho- (nº de conjuntos de itens LR(1) + nº de itens LR(1)) palavras

- Matriz GOTO para conjuntos LR(1)
  Tamanho- (nº de conjuntos de itens LR(1) x nº de símbolos) palavras

- Matriz para identificar se as produções de um não terminal com respectivos "lookaheads" foram adicionados ao fechamento de um conjunto de itens LR(1)
  Tamanho- ((Nº de não terminais-1) x (nº de terminais + 1)) "bits"

  Observação- Esta área por estar definida dentro da rotina que realiza o fechamento LR(1), só é ativada quando esta rotina é chamada.

Geração das tabelas LR(1):

- Vetor para identificar ações na tabela de análise sintática LR(1) para um estado qualquer
  Tamanho- (nº de terminais + 1) palavras.

6. Operações realizadas por rotina

6.1- Cálculo do FIRST da gramática de entrada

Para cálculo da relação FIRST utiliza-se uma variável lógica que indica a adição ou não de terminais ou ε à relação FIRST de algum não terminal no passo anterior. Percorre-se em cada passo os lados direitos das produções dos não terminais até encontrar ou não terminal que gere ε , e para cada não terminal utilizado, percorre-se todos os terminais adicionando-os à relação FIRST.

A ordem do tempo gasto não é maior que k x nº de símbolos do lado direito x nº de terminais onde k- nº de passos realizados no algorítmo.

6.2- Impressão da relação FIRST

Para esta impressão percorre-se todos os terminais para cada não terminal da gramática. Portanto a rotina processa em tempo O (nº de terminais x nº de não terminais).

6.3- Cálculo do FOLLOW da gramática de entrada

Para cálculo da relação FOLLOW utiliza-se de uma variável lógica que indica a adição ou não de terminais à relação no passo anterior. Percorre-se em cada passo os símbolos dos lados direitos das produções dos não terminais, e para cada símbolo percorre-se os demais a sua direita na produção até encontrar um terminal ou um não terminal que não gere ε. Para cada não terminal utilizado percorre-se todos os terminais adicionando-os à relação FOLLOW.

A ordem do tempo gasto não é maior que k x nº de símbolos do lado direitox (nº de símbolos do lado direito-nº de produções) x nº de terminais onde k - nº de passos realizados no algorítmo.

6.4- Impressão da relação FOLLOW

Idêntico ao FIRST (III.6.2).

6.5- Geração dos conjuntosde itens LR(0)

Percorre-se todos os conjuntos de itens realizando para cada um destes as operações:

- Fechamento deste conjunto de itens que é no máximo de ordem de tempo igual ao nº de itens centrais x nº de produções.

- Percorre-se todos os itens do fechamento gerando-se novos conjuntosde itens. Para cada conjunto de itens, ordena-se os seus itens centrais e verifica-se nos conjuntos de itens gerados anteriormente se existe algum igual a ele, o que acontece quando o nº de itens centrais é o mesmo e estes itens são iguais um a um.

6.6- Geração da tabela SLR(1)

Percorre-se todos os conjuntos de itens realizando para cada um destes as operações:

- Fechamento deste conjunto de itens que é no máximo de ordem de tempo igual ao nº de itens centrais x nº de produções.

- Para cada item de fechamento, se o ponto está na posição mais a direita da produção percorre-se todos os terminais para utilizar os de interesse.

6.7- Geração dos conjuntos de itens LALR(1)

Percorre-se todos os itens centrais LR(0) realizando para cada um destes as seguintes tarefas:

- Fechamento do item que é no máximo da ordem de tempo igual ao nº de produções x nº de símbolos do lado direito.

- Percorre-se todos os itens do fechamento. Se o item LALR(1) é um item gerado espontaneamente, insere-se este item no conjunto de itens LALR(1) respectivo, em ordem de itens se este não estiver lá. A inserção na pilha de trabalho do algorítmo e de item LR(0) com "lookahead" propagado é de ordem de tempo igual a uma constante.

- Para todos os itens gerados na pilha de trabalho , percorre-se os itens LR(0) com "lookahead" propagado do item LR(0) da pilha, inserindo-se estes itens no conjunto de itens LALR(1) em ordem de itens se estes não estiverem lá.

6.8- Geração da tabela LALR(1)

Para cada conjunto de itens LALR(1) realiza-se as seguintes tarefas:

- Fechamento LALR(1) que é da ordem de tempo no máximo igual ao (nº de itens centrais x nº de produções x nº de símbolos do lado direito).

- Para cada símbolo terminal verifica-se as ações em pilha.

- Percorre-se os itens do fechamento e verifica- se as ações reduza.

6.9- Geração dos conjuntos de itens LR(1)

Para cada conjunto de itens realiza-se as tarefas:

- Fechamento LR(1) deste conjunto de itens que é no

máximo de ordem de tempo igual ao nº de itens centrais x nº de produções x nº de símbolos do lado direito.

- Percorre-se todos os itens do fechamento gerando - se novos conjuntos de itens. Para cada conjunto de itens, ordena-se os seus itens centrais e verifica se nos conjuntos de itens gerados anteriormente se existe algum igual a ele, o que acontece quando o nº de itens é o mesmo e estes itens iguais um a um.

6.10-Geração da Tabela LR(1)

Idêntico ao (III.6.8) só que para LR(1).

7. Utilização de Macros

A utilização das estruturas nos algorítmos por tradução direta da forma em que estas foram implementadas, dificultaria o entendimento destes algorítmos. A linguagem ALGOL permite a criação de macros através do comando DEFINE que resolvem o problema acima mecionado.

Para se ter uma idéia do problema é dado um exemplo a seguir.

Se existe a necessidadede descobrir se um terminal T pertence ao FIRST do não terminal NT utiliza-se as estruturas

BOOLEAN ARRAY MATFIRST (1: nº de não terminais, 0: nº de palavras para armazenar nº de bits igual ao nº de terminais)

O acesso direto a estrutura seria da forma

IF MATFIRST (NT, ⌊(T - nº de não terminais-1)/48⌋.
(T - nº de não terminais- 48 x
⌊(T - nº de não terminais-1)/48)⌋))

onde a parte calculada após o • é a posição do bit na palavra à esquerda deste.

A utilização da macro FIRST definida como abaixo

```
DEFINE FIRST(NAOTERM,TERM)=

    MATFIRST(NAOTERM, ⌊(TERM-nº de não terminais-1)/48⌋.
                (TERM-nº de não terminais - 48 x
                ⌊(TERM-nº de não terminais -1)/48⌋ ))
```

O acesso a estrutura seria feito da seguinte maneira

```
IF FIRST(NT,T)
```

o que é muito mais intuitivo.

8. Resultados complementares FIRST e FOLLOW

Para a geração das tabelas de análise sintática SLR(1), LALR(1) e LR(1) foi necessário o cálculo da relação FIRST e FOLLOW para os símbolos não terminais desta gramática. Com o objetivo de deixar disponível o resultado destas relações para quem está interessado apenas no cálculo destas, implementou-se o gerador de forma a tornar isto possível (Ver capítulo V).

IV. DOCUMENTAÇÃO

1. Conversor da gramática de entrada em cartão para representação interna na memória

1.1- Descrição

O conversor executa o procedimento geral para análise sintática LR (II.7.3) utilizando-se da tabela LALR(1) da gramática (V.1.1.1), sendo que esta tabela contém ações para recuperação de erro e geração da representação interna da gramática de entrada. Um analisador léxico passa a este conversor sempre que necessário e um a um os símbolos codificados no arquivo CARD de entrada.

1.2- Descrição do Analisador Léxico

Os símbolos reconhecidos pelo analisador léxico e respectivas representações internas são as seguintes:

Representação interna	Descrição
1	espaços
2	=
3	;
4	|
5	identificador
6	cadeia
7	$(EOF)

As classes e respectivas representações internas para os caracteres possíveis nos cartões de entrada (arquivo CARD) são as seguintes:

Representação interna	Nome Mnemônico	Descrição
0	d	carater desconhecido
1	=	=
2	|	|
3	;	;
4	L≠Q	Letra ≠ Q
5	LQ	Letra Q
6	D	Digito
7	∅	branco
8	#	#
9	–	–
10	'	'
11	op	demais operadores
12		$(EOF)

(IV.2)

As operações realizadas nas transações no diagrama de estado do analisador léxico são as seguintes:

Representação interna	Nome Mnemônico	Descrição
1	ADD	Adicionar o carater lido ao vetor com o nome do símbolo CHARS
2	GETCHAR (GC)	Obter o próximo carater
3	GETFIRST (GF)	Obter o primeiro caracter do próximo cartão
4	ADDPLIC (ADDP)	Modificar o caracter # para ' no vetor com o nome do símbolo CHARS
5	LOOKUPTERM (LKT)	Pesquisar na área que contém o nome dos terminais se existe terminal com nome igual ao do vetor CHARS.Caso sim passa a representação interna do terminal, e caso não adiciona o nome deste terminal e gera a sua representação interna.
6	LOOKUPNAOTERM (LKNT)	idêntico ao 5 só que para não terminais
7	$=	Mover representação interna 2 para SIMBOL
8	$|	Mover representação interna 4 para SIMBOL
9	$;	Mover representação interna 3 para SIMBOL
10	$espaços	Mover representação interna 1 para SIMBOL

11	$cadeia	Mover representação interna 6 para SIMBOL
12	$identificador	Mover representação interna 5 para SIMBOL
13	$EOF	Mover representação interna 7 para SIMBOL
14	IM	Imprime mensagem "CARACTER DESCONHECIDO IGNORADO"
15	RC	Retira ultimo caracter do vetor CHARS

(IV.3)

São funções do analisador léxico:

- Ler e listar os cartões do arquivo CARD

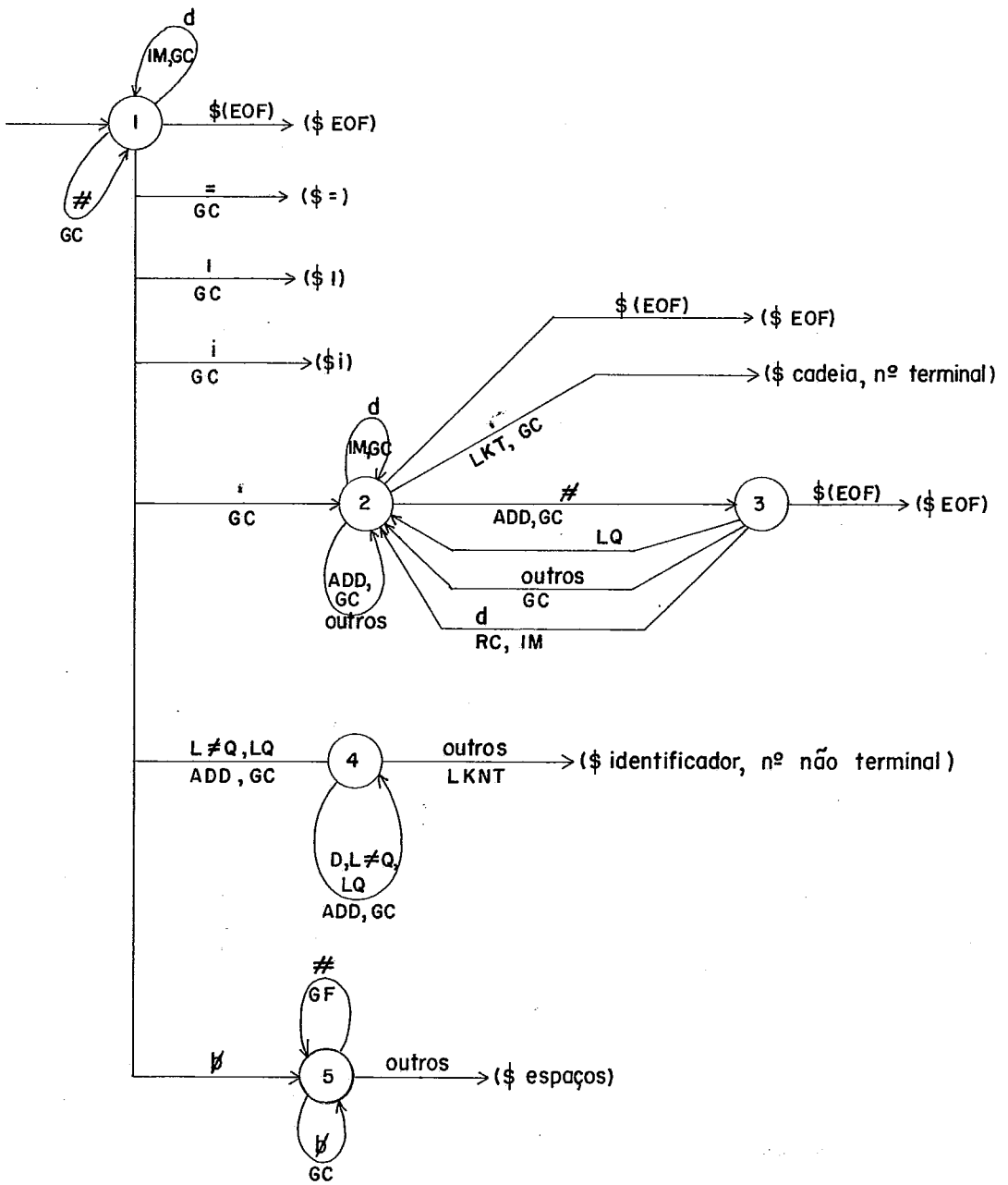
- Ignorar comentários

- Rejeitar caracteres ilegais e imprimir mensagens de erro

- Determinar os símbolos da linguagem de entrada

- Descobrir as representações internas e nome dos terminais (cadeias) e dos não terminais (identificadores) e representação interna dos outros símbolos passados ao analisador sintático.

O analisador léxico executa o diagrama de estado a seguir.


d
IM,GC
1
$(EOF)
($ EOF)
#
GC
=
GC
($ =)
!
GC
($!)
i
GC
($ i)
$(EOF)
($ EOF)
($ cadeia, nº terminal)
d
IM,GC
LKT, GC
2
GC
#
ADD,GC
3
$(EOF)
($ EOF)
LQ
ADD,
GC
outros
outros
GC
d
RC, IM
L≠Q,LQ
ADD, GC
4
outros
LKNT
($ identificador, nº não terminal)
D,L≠Q,
LQ
ADD, GC
#
GF
ø
5
outros
($ espaços)
ø
GC

1.2.1- Rotinas utilizadas

A rotina GETCHAR cuja função é obter o próximo caracter a ser analisado lexicamente, foi definida fora da rotina SCANNER de análise léxica devido a necessidade de chamada desta rotina dentro da rotina de conversão (CONVERSOR) para a leitura do primeiro caracter. São também funções da rotina GETCHAR; ler e listar cartões de entrada, obter o valor do próximo caracter e colocá-lo na variável CHAR, obter a representação interna da classe do caracter e colocá-lo na variável CLASSE, e mover o valor 12 para CLASSE se não existe mais cartão a ser lido.

As rotinas de análise léxica utilizam as seguintes variáveis mais importantes:

CARTAO-	contém o cartão recentemente lido
CHARARRAY-	variável de trabalho para permitir a operação ADD através de ponteiros
CHARS-	vetor que contém o nome símbolo terminal (cadeia) ou não terminal (identificador) lido.
MATNOMETERM-	vetor com nome dos terminais analisados
MATNOMENAOTERM-	vetor com nome dos não terminais analisados
CHAR-	contém a representação ebcdic do caracter obtido pela rotina GETCHAR
CLASSE-	contém a classe do caracter obtido pela rotina GETCHAR
COLUMN-	variável usada na rotina GETCHAR para descobrir a coluna do cartão em utilização
NUMNAOTER-	contém a representação interna do não terminal passada ao analisador sintático
NUMTER-	contém a representação interna do terminal passada ao analisador sintático

NCHAR- Se o símbolo é um identificador (não terminal) ou uma cadeia (terminal) indicará o número de caracteres deste.

SIMBOL- Contém a representação interna do símbolo passado para o analisador sintático.

MATESTSEGUINTE- Matriz que indica as transições dos estados do diagrama anteriormente apresentado

MATOPERACOES- Matriz que indica as operações realizadas nas transições dos estados do diagrama anteriormente apresentado

MATCLASSE- Matriz que contém a classe dos caracteres lidos o que é obtido utilizando-se como índice o valor decimal correspondete a representação hexadecimal deste caracter no código EBCDIC.

A rotina SCANNER quando é chamada começa no estado 1, consulta a matriz MATESTSEGUINTE para obter o próximo estado a partir da classe do caracter lido, e a matriz MATOPERACOES para as operações que deve realizar na transição entre estes dois estados. O processo é repetido até que se alcance o estado 6 que indica a existência de um símbolo a ser passado ao analisador sintático.

1.3- Descrição da análise sintática, recuperação de erro e geração de código

Esta etapa do conversor utiliza-se da tabela LALR(1) seguinte:

AÇÃO (operação, ação, cod.auxiliar)

Estado	Espaços	=	;	|	id	cadeia	$
0	(,e,2)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)
1	(, ,2)	(, , 2)	(, , 2)	(, , 2)	(1,e,6)	(, , 2)	(, a ,)
2	(, ,2)	(, , 2)	(, , 2)	(, , 2)	(, r,1)	(, ,)	(, r,1)
3	(,e,7)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)
4	(, ,3)	(,e, 8)	(, , 3)	(, , 3)	(, , 3)	(, , 3)	(, , 3)
5	(,e,9)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)
6	(,r,10)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)
7	(, ,2)	(, , 2)	(, , 2)	(, , 2)	(, r,2)	(, , 2)	(, r,2)
8	(,e,12)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)
9	(, ,3)	(,r, 6)	(, , 3)	(, , 3)	(, , 3)	(, , 3)	(, , 3)
10	(, ,4)	(, , 5)	(4,e,13)	(4,e,14)	(, ,)	(, ,)	(, , 5)
11	(, ,2)	(, , 2)	(,r,4)	(,r,4)	(2,e,6)	(3,e,17)	(, , 2)
12	(, ,5)	(, , 5)	(5,r,7)	(5,r,7)	(,r, 7)	(, r,7)	(, , 5)
13	(,r,3)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)
14	(,e,12)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)
15	(,e,19)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)
16	(,e,20)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)
17	(,r,11)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)	(, , 1)
18	(, , 2)	(, , 2)	(,r, 5)	(,r, 5)	(2,e,6)	(3,e,17)	(, , 2)
19	(, ,5)	(, , 5)	(,r, 8)	(,r, 8)	(,r, 8)	(,r, 8)	(, , 5)
20	(, ,5)	(, , 5)	(,r, 9)	(,r, 9)	(,r, 9)	(,r, 9)	(, , 5)

GOTO

Estado	CFG	R	ALT	LP	RP	NT	T
0	1						
1		3		4		5	
8			10		11		
11						16	15
14					18		
18						16	15

(IV.5)

Os conjuntos de itens LALR(1) associados a estados na tabela são:

0:	S →	•CFG	$
	CFG→	•espaços	$,id
	CFG→	•CFG R espaços	$,id
1:	CFG→	CFG•R espaços	$,id
	S →	CFG•	$
	R →	•LP = ALT ;	espaços
	LP →	•NT espaços	=
	NT →	•id	espaços
2:	CFG→	espaços•	$,id
3:	CFG→	CFG R•espaços	$,id
4:	R →	LP• = ALT ;	espaços
5:	LP →	NT• espaços	=
6:	NT →	id•	espaços
7:	CFG→	CFG R espaços•	$,id
8:	R →	LP = •ALT ;	espaços
	ALT→	•RP	;, |
	ALT→	•ALT|RP	;, |
	RP →	•espaços	;, |,id,cadeia
	RP →	•RP T espaços	;, |,id,cadeia
	RP →	•RP NT espaços	;, |,id,cadeia
9:	LP →	•NT espaços	=
10:	ALT→	ALT•| RR	;,|
	R →	LP = ALT• ;	espaços

11:	ALT	→	RP•	;,|
	RP	→	RP•NT espaços	;,cadeia,|, id
	RP	→	RP• T espaços	;,cadeia,|, id
	NT	→	•id	espaços
	T	→	•cadeia	espaços
12:	RP	→	espaços•	;,cadeia,|,id
13:	R	→	LP = ALT ;•	espaços
14:	ALT	→	ALT |•RP	|,;
	RP	→	•espaços	;,|,id,cadeia
	RP	→	•RP T espaços	;,|,id,cadeia
	RP	→	•RP NT espaços	;,|,id,cadeia
15:	RP	→	RP T•espaços	;,|,id,cadeia
16:	RP	→	RP NT•espaços	;,|,id,cadeia
17:	T	→	cadeia•	espaços
18:	RP	→	RP•T espaços	;,|,id,cadeia
	RP	→	RP•NT espaços	;,|,id,cadeia
	ALT	→	ALT| RP•	;,|
	NT	→	•id	espaços
	T	→	•cadeia	espaços
19:	RP	→	RP T espaços•	;,|,id,cadeia
20:	RP	→	RP NT espaços•	;,|,id,cadeia

A Matriz AÇÃO possui para cada elemento o terno (operação, ação, código auxiliar) que tem o seguinte significado:

operação- quando preenchido indica ação de características de geração de código que são as seguintes:

1- Salvar a representação interna do não terminal contido em SIMBOL e que corresponde ao lado esquerdo de uma produção.

2- Mover a representação interna de um não terminal do lado direito de uma produção para a área de memória que contém a gramática na representação utilizada pelo conversor.

3- Idêntico à operação 2 só que para um terminal.

4- Marcar indicação de final de produção.

5- Marcar indicação de existência de uma produção "epsilon".

ação: se estiver em branco indica existência de erro. Caso esteja preenchido deve conter um dos valores:

e- indica ação"empilha"

r- indica ação "reduza"

a- indica ação "aceita".

código auxiliar- tem seu siginificado associado a ação na seguinte forma:

Ação	Descrição
empilha	indica o estado que passa ao topo da pilha

Ação	Descrição
reduza	indica a produção utilizada para a redução
aceita	sem significado
branco	indica a ação de recuperação de erro realizada que pode ser uma das se - guintes:

1-imprime mensagem ESPACO AUSENTE e insere o símbolo espaço como próximo símbolo na sentença de entrada.

2-idêntico ao código 1 para símbolo identificador.

3-Idêntico ao código 1 para símbolo =

4-idêntico ao código 1 para símbolo |.

5-idêntico ao código 1 para símbolo ;

São funções desta fase do conversor:

- Identificar e recuperar erros na sentença de entrada
- Reconhecer uma sentença de entrada válida
- Gerar representação na memória da gramática de entrada codificada no arquivo CARD
- Imprimir relatórios com as representações internas dos não terminais e terminais com respectivos nomes.
- Identificar a existência de erro grave na análise sintática da sentença de entrada que impedirá posterior geração das tabelas SLR(1), LALR(1) e LR(1).

As variáveis mais importantes utilizadas são:

SIMB- contém a representação numérica do símbolo correntemente analisado, que pode ter sido obtido por recuperação de erro ou pelo analisador léxico.

HOUVERECUPERRO- indica se ocorreu recuperação de erro ou não.

MATACTION- contém as ações da tabela de análise sintática LALR(1) (IV.5)

MATACTIONOPER- contém as operações da tabela de análise sintática LALR(1) (IV.5)

MATACTIONCAUX- contém os códigos auxiliares da tabela de análise sintática LALR(1) (IV.5)

MATGOTO- contém os resultados da função GOTO da tabela de análise sintática LALR(1) (IV.5)

MATPROD- contém os símbolos não terminais do lado esquerdo das produções e também tamanho destas para a gramática que originou a tabela LALR(1) (IV.5)

STACKCITEM- área utilizada para empilhar os estados na análise sintática LALR(1).

STACKSIMB- área utilizada para empilhar os símbolos na análise sintática LALR(1).

As rotinas utilizadas nesta fase são:

REALIZAOPERACAO- chamada pelo conversor para realizar o procedimento relativo ao código de operação encontrado ao percorrer a tabela LALR(1) (IV.5) na análise sintática da sentença de entrada.

SHIFT- chamada pelo conversor quando a tabela LALR(1) (IV.5) indica uma ação "empilha" na análise sintática da sentença de entrada. O procedimento realizado é o correspondente a ação "empilha" do algorítmo padrão para análise sintática LR. (II.7.3).

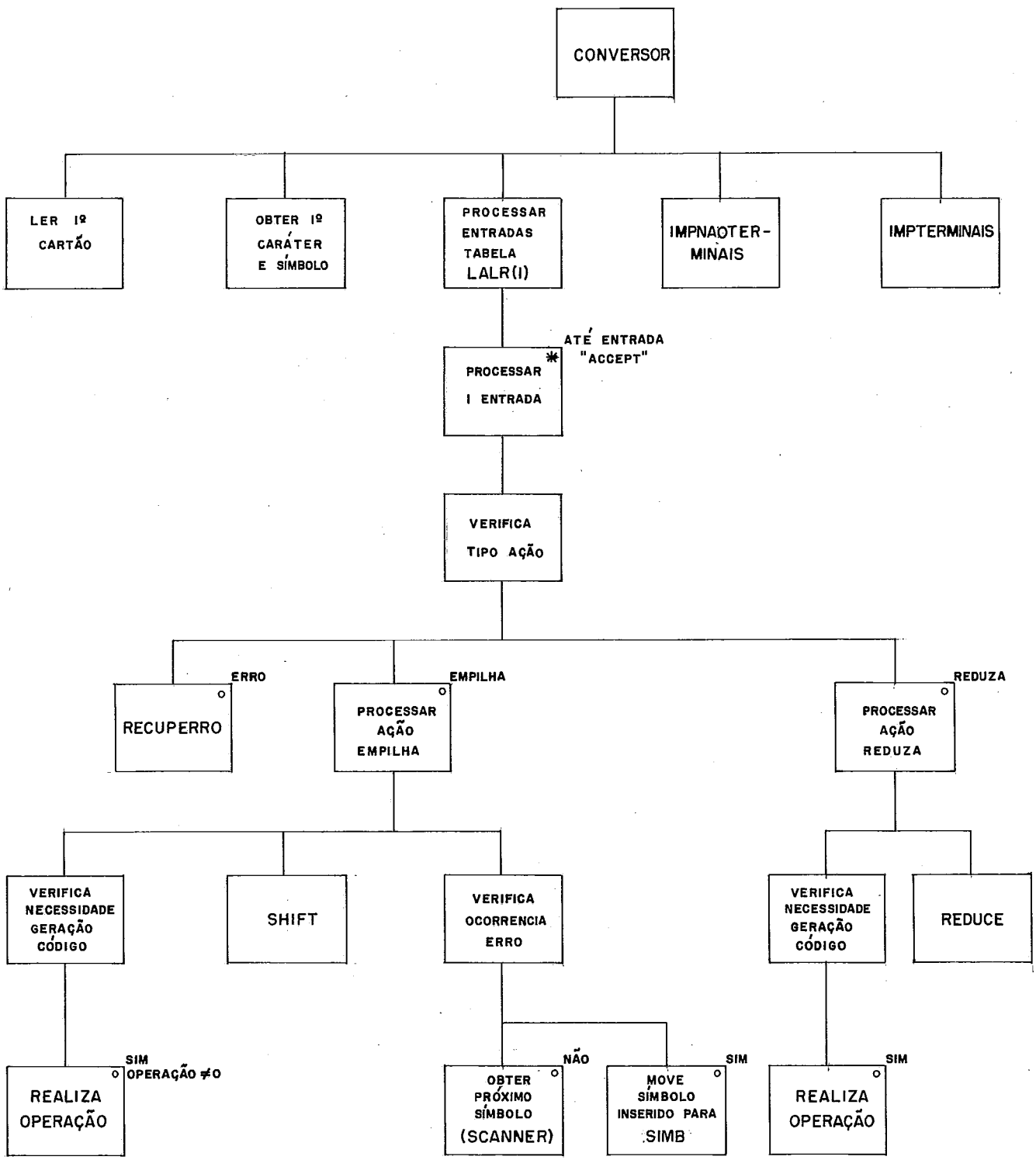
REDUCE- idêntico a rotina SHIFT para ação "reduza".

RECUPERRO- chamada pelo conversor sempre que este encontra um valor igual a zero para o código de ação na tabela na análise sintática LALR(1). O procedimento de recuperação de erro está codificado no campo de código auxiliar,e as suas funções já foram mencionadas anteriormente.

IMPNAOTERMINAIS- chamada pela rotina CONVERSOR após a análise sintática da sentença de entrada. Esta rotina percorre o vetor MATNOMENAOTERM e imprime relatório com a representação interna dos não terminais e respectivos nomes.

IMPTERMINAIS- idêntico a rotina IMPNAOTERMINAIS só que para terminais e o vetor MATNOMETERM.

1.4- Fluxo


CONVERSOR
LER 1º CARTÃO
OBTER 1º CARÁTER E SÍMBOLO
PROCESSAR ENTRADAS TABELA LALR(I)
IMPNAOTER-MINAIS
IMPTERMINAIS
PROCESSAR I ENTRADA
*
ATÉ ENTRADA "ACCEPT"
VERIFICA TIPO AÇÃO
ERRO
RECUPERRO
EMPILHA
PROCESSAR AÇÃO EMPILHA
REDUZA
PROCESSAR AÇÃO REDUZA
VERIFICA NECESSIDADE GERAÇÃO CÓDIGO
SHIFT
VERIFICA OCORRENCIA ERRO
VERIFICA NECESSIDADE GERAÇÃO CÓDIGO
REDUCE
SIM
OPERAÇÃO ≠0
REALIZA OPERAÇÃO
NÃO
OBTER PRÓXIMO SÍMBOLO (SCANNER)
SIM
MOVE SÍMBOLO INSERIDO PARA SIMB
SIM
REALIZA OPERAÇÃO

2. Rotinas de cálculo da relação FIRST para a gramática de entrada

2.1- Descrição geral

Estas rotinas são duas: a rotina GERAFIRST que calcula a relação FIRST para os símbolos não terminais da gramática,e a rotina IMPFIRST que imprime a tabela resultado desta relação. A rotina IMPFIRST só é chamada se for solicitado o procedimento FIRST no cartão de entrada PROCED, e a rotina GERAFIRST se solicitados os procedimentos FIRST,FOLLOW ou LR1.

2.2- Rotina GERAFIRST

A relação é calculada apenas para os símbolos não terminais aplicando-se para todas as regras 2 e 3 descritas em (II.7.4.5.1) até que não possam ser adicionados terminais ou ε ao FIRST destes não terminais.

As variáveis mais importantes utilizadas são:

MATGERAEPSILON- contém indicação se ε pertence ao FIRST de um não terminal qualquer. É implementada através de um vetor de "bits".

MATFIRST- contém indicação se um terminal pertence ao FIRST de um não terminal qualquer.É implementada através de uma matriz de "bits" com índice igual a representação interna do não terminal e a linha contém todos os terminais.

As rotinas auxiliares utilizadas são :

ADDFIRST- adiciona o terminal passado como parâmetro à relação FIRST do símbolo não terminal também passado como parâmetro.

ADDFIRSTEPSILON- adiciona ε ao FIRST do não terminal passado como parâmetro na matriz MATGERAEPSILON.

ADDLINHAFIRST- Adiciona à relação FIRST do não terminal NAOTERM2 passado como parâmetro, todos os terminais pertencentes a relação FIRST do não terminal NAOTERM1 também passado como parâmetro. É implementado através da operação OR entre os "bits" de duas linhas na Matriz MATFIRST.

2.3- Rotina IMPFIRST

Percorre a Matriz MATFIRST para cada não terminal e imprime a lista de terminais que pertencem a relação FIRST deste. Percorre também a matriz MATGERAEPSILON e imprime uma linha com a palavra EPSILON se ε pertence ao FIRST deste não terminal. O leiaute do relatório está em (ANEXO II) e são impressas somente as representações internas dos símbolos.

3. Rotinas de cálculo da relação FOLLOW para a gramática de entrada

3.1- Descrição Geral

Estas rotinas são duas: a rotina GERAFOLLOW que calcula a relação FOLLOW para os símbolos não terminais da gramática de entrada, e a rotina IMPFOLLOW que imprime a tabela resultado desta relação. A rotina IMPFOLLOW só é chamada se for solicitado o procedimento FOLLOW no cartão de entrada PROCED, e a rotina GERAFOLLOW se solicitados os procedimen-

tos FOLLOW ou SLR:

3.2- Rotina GERAFOLLOW

A relação é calculada aplicando-se as regras 2 e 3 descritas em (II.7.4.5.2) até que possam ser adicionados terminais ou $ ao FOLLOW destes não terminais.

A regra 1 é aplicada nas inicializações de cálculo.

Variáveis mais importantes:

MATFOLLOW- contém indicação se um terminal ou $, pertence ao FOLLOW de um não terminal qualquer. É implementada através de uma matriz de "bits" com índice igual a representação interna do não terminal e a linha contém todos os terminais.

As rotinas auxiliares utilizadas são:

ADDFOLLOW- adiciona o terminal ou $ passado como parâmetro ao FOLLOW do não terminal , também passado como parâmetro na Matriz MATFOLLOW.

ADDLINHAFOLLOW- Adiciona à relação FOLLOW do não terminal NAOTERM2 passado como parâmetro, todos os terminais pertencentes à relação FOLLOW do não terminal NAOTERM1 também passado como parâmetro. É implementada através da operação OR entre os "bits" de duas linhas na matriz MATFOLLOW.

ADDLINHAFIRSTFOLLOW- Adiciona à relação FOLLOW do não terminal NAOTERM2 passado como parâmetro,todos os terminais pertencentes

a relação FIRST do não terminal NAOTERM1 também passado como parâmetro. É implementada através da operação OR entre os "bits" de uma linha da matriz MATFIRST com outra da matriz MATFOLLOW.

3.3- Rotina IMPFOLLOW

Percorre a matriz MATFOLLOW para cada não terminal e imprime a lista de terminais pertencentes a relação FOLLOW deste. O leiaute do relatório está em (ANEXO II) e são impressas somente as representações internas dos símbolos.

4. Rotinas de uso geral

4.1- Rotina IMPRODUCOES

Percorre a matriz MATPRODUCAO que faz parte da representação na memória da gramática de entrada,e imprime as produções da gramática com as representações internas dos símbolos respectivos. O leiaute do relatório está em (ANEXO II) e este só é impresso se não houve erro grave na análise sintática da sentença de entrada.

4.2- Rotina ELIMINACLOS

Tem a função de eliminar um item passado como parâmetro, da área utilizada para cálculo do FECHAMENTO de um conjunto de itens LR(0) ou LR(1). Só é modificada a área com os ponteiros para próximo item e item anterior.

5. Rotinas de cálculo das tabelas de análise sintática SLR(1) e LALR(1)

5.1- Descrição Geral

Para cálculo das tabelas SLR(1) e LALR(1) utilizou-se uma rotina de nome SLRLALR chamada pela rotina GERPARSER sem

pre que forem solicitados no cartão de entrada PROCED os procedimentos SLR ou LALR1.

A rotina SLRLALR chama inicialmente a rotina GERCONJ ITEMLRO para cálculo dos conjuntos de itens LR(0) e matriz da função GOTO para estes conjuntos de itens.

A seguir chama-se a rotina IMPGOTOLRO para impressão da função GOTO.

Se o procedimento SLR foi solicitado no cartão de entrada PROCED, chama-se a rotina GERACTIONSLR para obtenção da tabela SLR(1).

Se o procedimento LALR1 foi solicitado no cartão de entrada PROCED, chama-se a rotina LALR para cálculo dos conjuntos de itens LALR(1) pela rotina GERCONJITEMLALR1 e obtenção da tabela LALR(1) pela rotina GERACTIONLALR1.

5.2- Rotina CLOSURELRO

Utilizada para realizar o FECHAMENTO do conjunto de itens LR(0) passado como parâmetro com o nome CITEM.

O algorítmo descrito em (II.7.4.2) é implementado com uma fase inicial de adição dos itens centrais pertencentes ao conjunto de itens CITEM de entrada, na área de memória STACKCLOS. A fase final consiste em aplicar o algorítmo propriamente dito adicionando itens a partir de itens adicionados.

Variáveis importantes utilizadas:

ADD- Vetor de "bits" indicando se os itens relativos as produções de um não terminal já foram adicionados ao FECHAMENTO.

STACKCLOS- Pilha de armazenamento dos itens de fechamento . Ver (III.3.4)

5.3- Rotina GERCONJITEMLRO

Implementa algorítmo apresentado em (II.7.4.4) chamando a rotina GERAGOTO para cada conjunto de itens não analisado. A rotina GERAGOTO tem por função adicionar em área na memória todos os conjuntos de itens gerados a partir do conjunto de itens em análise que é passado como parâmetro.

Variáveis importantes utilizadas:

MATCONJITEM,MATITEM- Contém os conjuntos de itens LR(0). Ver (III.3.2).

MATGOTO - Matriz que contém o resultado da função GOTO(I,X) onde I é um conjunto de itens LR(0) e X um símbolo da gramática de entrada.

Rotinas auxiliares utilizadas:

GERAGOTO- Calcula inicialmente o FECHAMENTO do conjunto de itens passado como parâmetro. Para o primeiro item do FECHAMENTO que tenha símbolo à direita do ponto e a partir deste para os itens que tenham este mesmo símbolo à direita do ponto, insere-se estes itens na área que contém os conjuntos de itens. Se não existe símbolo à direita do ponto para o item analisado, e para cada item inserido no processo anterior, estes devem ser eliminados da área de FECHAMENTO através de chamada à rotina ELIMINACLOS para evitar reanalisá-los na próxima utilização da área STACKCLOS. Es-

te processo continua até que a área STACKCLOS esteja vazia. A cada geração de um novo conjunto itens chama-se a rotina PROCURACITEMIDENTICO para verificar se este conjunto de itens foi gerado anteriormente, caso sim libera-se a área utilizada para a sua geração.

PROCURACITEMIDENTICO- Verifica se um conjunto de itens passado como parâmetro já foi gerado anteriormente. Para facilitar a busca, inicialmente ordena-se na memória os itens gerados.

5.4- Rotina IMPGOTOLRO

Percorre a matriz MATGOTO para cada conjunto de itens e imprime, se existir pelo menos um resultado para a função GOTO deste conjunto de itens, uma linha com a lista de símbolos e conjuntos de itens resultados desta função. Para detalhes de leiaute ver (ANEXO.II).

5.5- Rotina GERACTIONSLR

Implementa o algorítmo descrito em (II.7.4.6). Esta rotina é também utilizada para imprimir os conjuntos de itens LR(0), os conflitos existentes na geração da tabela SLR(1), e a tabela de análise sintática SLR(1). Ver leiautes respectivos em (ANEXO.II).

Variáveis auxiliares importantes:

ACTIONSLR- Vetor com indicação das ações na tabela de análise sintática SLR(1) para o conjunto de itens correntemente analisado, e tem por função descobrir os tipos de conflitos existentes.

Rotinas auxiliares utilizadas:

CABECALHOPSLR- Utilizada para imprimir cabeçalho do relatório da tabela de análise sintática SLR (1).

CABECALHOCLRO- Utilizada para imprimir cabeçalho do relatório com os conjuntos de itens LR(0).

INPCONFLITOSLR- Chamada para imprimir uma linha do relatório com os conflitos da tabela de análise sintática SLR(1),e se necessário chama rotina interna de cabeçalho deste relatório.

ADDACTIONSLR- Chamada pela rotina GERACTIONSLR sempre que é necessário adicionar uma entrada na tabela de análise sintática SLR(1). A rotina verifica na matriz ACTIONSLR se já houve alguma entrada anterior para o símbolo e conjunto de itens em análise. Caso sim, chama a rotina IMPCONFLITOSLR e caso não, move a entrada para matriz ACTIONSLR e para área de trabalho que quando totalmente preenchida deve ser impressa no relatório da tabela de análise sintática SLR(1).

5.6- Rotina LALR

5.6.1- Descrição

Parte dos conjuntos de itens LR(0) e calcula os conjuntos de itens LALR(1) pela rotina GERCONJITEMLALR1 e a seguir calcula a tabela de análise sintática LALR(1) a partir dos conjuntos de itens LALR (1).

5.6.2- Rotina GERCONJITEMLALR1

Implementa o algorítmo apresentado em (II.7.6.2.3).

Na fase inicial para todos os itens centrais dos conjuntos de itens LR(0), calcula-se o FECHAMENTO do item LALR(1) formado com a produção e posição do ponto do item LR(0) e "lookahead" fictício #, inserindo-se na pilha de trabalho do algorítmo através da rotina INSERE os itens LALR(1) gerados espontaneamente, que são os que possuem "lookahead" diferente de # no FECHAMENTO anterior. O "lookahead" e a produção do item LALR(1) gerado são iguais às do item do FECHAMENTO e a posição do ponto deste mais um. Também são inseridos em uma outra área de memória através da rotina PROPAGA, os itens LR(0) que terão "lookaheads" propagados do item LR(0)base do FECHAMENTO, que são os que possuem "lookahead" iguais a #.

Na fase final utiliza-se os itens colocados na pilha de trabalho anteriormente e os colocados nesta fase. Para cada item da pilha insere-se nesta todos os itens LR(0) com "lookahead "propagado a partir da parte LR(0) deste item.

A representação dos conjuntos de itens LALR(1) adotada em (III.3.3) foi necessária devida as características de geração item a item do algorítmo e não a nível de conjunto itens como é o caso do SLR(1) e LR(1).

Variáveis importantes utilizadas:

LR1STKCITEM,LR1STKITEM- Pilhas de trabalho com os conjuntos de itens e itens LALR(1) gerados pelo algorítmo.

MATPROPAG- Contém a posição do primeiro item LR(0) que terá "lookahead" propagado do item LR(0) com produção e posição do ponto funcionando como índices desta matriz.

MATITEMLROPROPAG- Contém a lista de itens LR(0) mencionados na matriz anterior.

Rotinas auxiliares utilizadas:

CLOSURELR1ITEM- Realiza o FECHAMENTO do item LALR(1) formado com a produção (PRODITEM) e posição do ponto (POSPONTOITEM) passados como parâmetros e o "lookahead" #.

PROPAGA- Insere um item LR(0) que terá "lookahead" propagado de um outro item LR(0).

INSERT- Insere um item LALR(1) na área que contém os conjuntos de itens LALR(1).

5.6.3- Rotina CLOSURELALR1

Utilizada para realizar o FECHAMENTO de um conjunto de itens LALR(1) passado como parâmetro com o nome CITEM.

Implementa o algorítmo descrito em (II.7.5.2) que só é diferente da rotina CLOSURELR1 pelo fato de que a representação utilizada para os conjuntos de itens LALR(1) é diferente dos conjuntos de itens LR(1).

5.6.4- Rotina GERACTIONLALR1

Implementa o algorítmo descrito em (II.7.5.5) só que para os conjuntos de itens LALR(1). Esta rotina é também utilizada para imprimir os conjuntos de itens LALR(1), os conflitos existentes na geração da tabela LALR(1) e a tabela LALR(1). Ver leiautes em (ANEXO.II).

A variável ACTIONLALR e rotinas CABECALHOPLALR1, CABECALHOCLALR1, IMPCONFLITOLALR1, ADDACTIONLALR têm utilização idêntica às respectivas variáveis e rotinas na rotina GERACTIONSLR(IV.5.5), só que referentes a tabela e conjuntos de itens LALR(1).

6. Rotinas de cálculo das tabelas de análise sintática LR(1)

6.1- Descrição

A rotina que faz o cálculo das tabelas de análise sintática LR(1) tem o nome LR1. É chamada pela rotina GERPARSER sempre que solicitado no cartão de entrada PROCED o procedimento LR1.

A rotina LR1 chama a rotina GERCONJITEMLR1 para cálculo dos conjuntos de itens LR(1), a rotina IMPGOTOLR1 para impressão do resultado da função GOTO para estes conjuntos de itens, e a rotina GERACTIONLR1 para cálculo da tabela LR (1).

6.2- Rotina CLOSURELR1

Utilizada para realizar o fechamento de um conjunto de itens LR(1) passado como parâmetro com o nome CITEM.

O algorítmo descrito em (II.7.5.2) é implementado com uma fase inicial de adição dos itens centrais pertencentes

ao conjunto de itens CITEM na área de memória STACKCLOS

A fase final consiste em aplicar o algorítmo propriamente dito adicionando itens a partir dos já adicionados.

Variáveis importantes utilizadas:

MATADD- Matriz de "bits" indicando se os itens relativos as produções de um não terminal e o terminal "lookahead" já foram adicionados ao fechamento.

STACKCLOS- Pilha de armazenamento dos itens do fechamento . Ver (III.3.4).

Rotinas auxiliares:

ADDCLOS- Utilizada para adicionar itens LR(1) formado com as produções do não terminal NAOTERM passado como parâmetro,posição do ponto igual a zero e "lookahead" igual ao terminal TERM também passado como parâmetro.

6.3- Rotina GERCONJITEMLR1

Implementa algorítmo apresentado em (II.7.5.4) chamando a rotina GERAGOTO para cada conjunto de itens não analisado.

A rotina GERAGOTO tem por função adicionar em área na memória, todos os conjuntos de itens gerados a partir do conjunto de itens em análise que é passado como parâmetro.

Variáveis importantes utilizadas:

MATCONJITEM,MATITEM- Contém os conjuntos de itens LR(1). VER (III.3.2).

MATGOTO- Matriz que contém o resultado da fun - ção GOTO(I,X) onde I é um conjunto de itens LR(1) e X um símbolo da gramática de entrada.

Rotinas auxiliares utilizadas

GERAGOTO,PROCURACITEMIDÊNTICO- idênticas às utilizadas na rotina GERCONJITEMLRO, só que trabalham com conjuntos de itens LR(1).

6.4- Rotina IMPGOTOLR1

Idêntica à rotina IMPGOTOLRO só que trabalha com conjuntos de itens LR(1).

6.5- Rotina GERACTIONLR1

Implementa o algorítmo descrito em (II.7.5.5). É também utilizada para imprimir os conjuntos de itens LR(1), os conflitos existentes na geração da tabela LR(1), a tabela de análise sintática (LR1) [illegible]. Ver leiautes em (ANEXO. II).

A variável ACTIONLR e rotinas CABECALHOPLR1, CABECALHOCLR1, IMPCONFLITOLR1, ADDACTIONLR têm utilização idêntica as respectivas variáveis e rotinas na rotina GERACTIONSLR (IV.5.5), só que referentes a tabela e conjunto de itens LR(1).

V. UTILIZAÇÃO

1. Descrição de Arquivos

1.1- Entradas:

1.1.1- Arquivo CARD

Contém a gramática a ser utilizada para a geração de tabelas dos analisadores sintáticos SLR(1),LALR (1) e LR(1), escrita de acordo com a gramática para escrever gramáticas descritas adiante, e codificada em cartões em formato livre com as posições de 1 a 80 utilizáveis. Deve-se observar que não existe a necessidade de utilização das colunas 73 a 80 para sequenciamento de cartões, pois pode-se utilizar o símbolo de comentário #, fazendo-se que todo o cartão seja ignorado da coluna em que o # se posiciona até a coluna 80.Para simular o sequenciamento dos cartões coloca-se o símbolo # na coluna 73 e numera-se da coluna 74 a 80.

Gramática para descrição de gramáticas:

Parte tratada pela análise sintática:

<Gramática livre de contexto>:= espaços
|<Gramática livre de contexto> <Regra> espaços

<Regra > := < Parte esquerda > =< Alternativas > ;

<Alternativas> : = <Parte direita> |
<Alternativas>| <Parte direita >

<Parte esquerda>:= <Não terminal> espaços
<Parte direita>: = espaços
|<Parte direita><Terminal> espaços
|<Parte direita><Não Terminal> espaços
<Não Terminal>: = identificador
<Terminal>: = cadeia

Parte tratada pela análise léxica:

< Identificador>: = <Letra>
|<Identificador> <Letra >
|<Identificador> <Dígito>
|<Identificador> —

< Cadeia >: = ' <Qualquer caracter> '

< Espaços> : =<Separador>
| <Espaços> <Separador>

< Separador>: = branco

< Qualquer caracter>: = ε
|<Qualquer caracter><Caracter>
|<Qualquer caracter> # Q
|<Qualquer caracter> # #

< Caracter >: = <Letra> |<Dígito> |<Operador>

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

< Dígito> : = 0|1|2|3|4|5|6|7|8|9

< Operador>: = [|.|<|(|+|||&|]|$|*|)|;|¬|-|/|,|%|_|@|?|:|>|=|"

Observações:

- As produções <Qualquer caracter> : =< Qualquer caracter> # Q
e <Qualquer caracter> : =< Qualquer caracter> # #,
são utilizadas pelo analisador léxico da seguinte maneira:

Q é sempre transformado no caracter '
é sempre transformado no caracter

- Os comentários não são passados para análise sintática, tudo funciona como se o cartão acabasse antes da coluna 80.

- A produção <Qualquer caracter> : = ε não deve ser utilizada

quando a cadeia em formação é de tamanho zero, pois neste caso se estaria trabalhando com um terminal sem descrição correspondente. Esta situação não deve ser confundida com a descrição de uma produção ε que é obtida com as seguintes produções:

<Regra> := <Parte esquerda> = <Alternativas> ;

<Alternativas> : = <Parte direita>
<Parte direita> : = espaços

- O primeiro símbolo não terminal, parte esquerda da primeira produção, deve ser o não terminal S do qual se origina qualquer árvore de análise sintática desta gramática, pois existe a criação automática da produção S'→S, onde S' recebe a representação interna numérica de valor 0.

- A parte esquerda <Alternativas> tem apenas duas produções e não três como apresenta. O segundo | faz parte da segunda produção.

1.1.2- Cartão parâmetro PROCED

Cartão obrigatório que indica os tipos de procedimentos desejados da seguinte maneira:

Coluna	Procedimento desejado	Nome para Referência	Valor
1	relação FIRST	FIRST	Preencher com qualquer caracter caso desejado
2	relação FOLLOW	FOLLOW	Idem
3	tabelas de analisador sintático SLR(1)	SLR	Idem

4	tabelas de analisador sintático LALR(1)	LALR1	Idem
5	tabelas de analisador sintático LR(1)	LR1	Idem

Existem 13 outros tipos de procedimentos relacionados abaixo:

Código	Descrição	Nome para Referência
1	Cálculo da Relação FIRST	FIRST
2	Impressão de relatório com resultados da relação FIRST	IMPFIRST
3	Cálculo da Relação FOLLOW	FOLLOW
4	Impressão de relatório com resultados da relação FOLLOW	IMPFOLLOW
5	Cálculo dos conjuntos de itens LR(0)	GERCONJLRO
6	Impressão de relatórios dos conjuntos de itens LR(0)	IMPCONJLRO
7	Cálculo dos conjuntos de itens LALR(1)	GERCONJLALR1
8	Impressão de relatório dos conjuntos de itens LALR(1)	IMPCONJLALAR1
9	Geração e impressão do relatórios com a tabela do analisa - dor sintático SRL(1)	TABELASLR
10	Geração e impressão de relatório com a tabela do analisador sintático LALR(1)	TABELALALR1
11	Cálculo dos conjuntos de itens LR(1)	GERCONJLR1
12	Impressão de relatório dos conjuntos de itens LR(1)	IMPCONJLR1

13 Geração e impressão de relatório TABELALR1 com a tabela do analisador sintático LR(1).

Estes 13 procedimentos estão relacionados com os 5 procedimentos do cartão PROCED de forma indicada pela seguinte tabela:

Procedimentos	Lista de procedimentos associados												
Cartão PROCED	1	2	3	4	5	6	7	8	9	10	11	12	13
1	X	X											
2	X		X	X									
3	X		X		X	X			X				
4	X				X		X	X		X			
5	X										X	X	X

O gerador monta e executa o vetor resultado da união de procedimentos associados a cada procedimento do cartão PROCED desejado.

Exemplo:

Cartão PROCED:

Coluna	Valor	Referência
1	X	FIRST
2	branco	FOLLOW
3	X	SLR
4	branco	LALR1
5	X	LR1

Vetor montado

1	2	3	4	5	6	7	8	9	10	11	12	13
X	X	X		X	X			X		X	X	X

Serão executados os procedimentos assinalados com X.

1.1.3- Cartão parâmetro PARMCONV

Cartão opcional com parâmetro utilizado pelo conversor da gramática de entrada para dimensionar o número máximo de "bytes" utilizados para conter os nomes dos não terminais ou dos terminais. Cada nome gasta um número de "bytes" igual ao número de caracteres deste e mais um para separação com o próprio nome.

Se este cartão não estiver presente é assumido o valor de 40000 para a área de nomes dos símbolos não terminais ou terminais.

1.1.4- Cartão Parâmetro PARMGERAL

Cartão opcional com parâmetro geral utilizado para as gerações de tabelas (SLR(1) LALR(1), LR(1)) e deve conter em forma livre o tamanho máximo de número de itens necessários para realizar o fechamento de um conjunto de itens.

Se este cartão não estiver presente é assumido um tamanho máximo de número de itens igual a 10000.

1.1.5- Cartão Parâmetro PARMLRO

Cartão opcional com parâmetros utilizados pelos geradores de tabelas SLR(1) e LALR(1) contendo em forma livre na ordem, o total de conjuntos de itens

LR(0) e o total de itens centrais de todos os conjuntos LR(0).

Se este cartão não estiver presente são assumidos os valores de 2000 para o total de conjuntos de itens e 10000 para o total de itens.

1.1.6- Cartão Parâmetro PARMLALR

Cartão opcional com parâmetros utilizados pelo gerador de tabela LALR(1) contendo em forma livre o total de itens centrais LALR(1) de todos os conjuntos de itens LALR(1).

Se este cartão não estiver presente é assumido o valor de 10000 para o total acima.

1.1.7- Cartão Parâmetro PARMLR1

Cartão opcional com parâmetros utilizados pelo gerador de tabela LR(1) contendo em forma livre na ordem, o total de conjuntos de itens LR(1) e o total de itens centrais de todos os conjuntos de itens LR(1).

Se este cartão não estiver presente são assumidos os valores de 10000 para o total de conjunto de itens e 50000 para o total de itens.

1.1.8- Cartão Parâmetro PARMGRAM

Cartão opcional com parâmetros de carater geral que são utilizados para dimensionamento de áreas usadas para conversão da gramática de representação montada pelo conversor,para representação utilizada pelos geradores de tabelas SLR(1), LALR(1) e LR(1). Contém na ordem o número máximo de símbolos não terminais da gramática de entrada e o número máximo de símbolos pertencentes as partes direitas das produções.

Se este cartão não estiver presente são assumidos os valores de 300 para o número de não terminais e 2500 para os símbolos do lado direito das produções.

1.2- Saídas

1.2.1- Relatório da Gramática na forma de entrada e mensagens de erro

Contém as imagens dos cartões utilizados para entrada da gramática(arquivo CARD), com mensagens de erro relativas a análise sintática destes.

Observar que as mensagens relativas a um determinado cartão são impressas imediantamente antes da impressão deste.

Este relatório é sempre impresso.

1.2.2- Relatório de Representação Interna dos Não Terminais

Contém as representações internas numéricas de todos os símbolos não terminais da gramática de entrada e seus respectivos nomes.

Estes números são utilizados por todos os outros relatórios que contenham não terminais, não sendo feito mais referência aos nomes destes.

Este relatório é sempre impresso.

1.2.3- Relatório de Representação Interna dos Terminais

Contém as representações internas numéricas de todos os símbolos terminais e seus respectivos nomes.

Estes números são utilizados por todos os outros relatórios que contenham terminais não sendo feita mais referência aos nomes destes.

O último terminal ($) não pertence a gramática de entrada mas é importante nas tabelas de análise sintática LR, e indica o fim da sentença a ser escrita da gramática de entrada.

Este relatório é sempre impresso.

1.2.4- Relatório das Produções com códigos de representação interna

Contém todas as produções impressas com os códigos numéricos de representações de símbolos terminais e não terminais.

Este relatório é sempre impresso.

1.2.5- Relatório da relação FIRST

Contém para cada não terminal X todos os terminais que pertençam ao FIRST(X).

Este relatório só é impresso se a coluna 1 do cartão PROCED estiver preenchida e não tenha ocorrido erro grave na análise sintática da gramática de entrada.

1.2.6- Relatório da relação FOLLOW

Contém para cada não terminal X todos os terminais pertencentes ao FOLLOW(X).

Este relatório só é impresso se a coluna 2 do cartão PROCED estiver preenchida e não tenha ocorrido erro grave na análise sintática da gramática de entrada.

1.2.7- Relatório da função GOTO (LR(0))

Contém a função GOTO (I,X) para cada conjunto de itens I, listando os pares (símbolo X da gramática para os quais esta função existe, conjunto de itens resultados desta função).

Observar que a função GOTO(I,X) onde I é um conjunto de itens LR(0) é a mesma que quando I é um conjunto de itens LALR(1).

Este relatório só é impresso se as colunas 3 ou 4 do cartão PROCED estão preenchidas (SLR e LALR1), e não houve erro grave na análise sintática da gramática de entrada.

1.2.8- Relatório de conjunto de itens LR(0)

Contém para todo conjunto de itens LR(0) todos os itens LR(0) a este pertencentes. Um item LR(0) é representado por um par (produção, posição ponto).

Este relatório só é impresso se a coluna 3 (SLR) do cartão PROCED estiver preenchida e não houve erro grave na análise sintática da gramática de entrada.

1.2.9- Relatório de conjuntos de itens LALR(1)

Contém para todos os conjuntos de itens LALR(1) todos os itens LALR(1) a este pertencentes. Um item LALR(1) é representado por um terno (produção, posição ponto, terminal "lookahead").

Este relatório só é impresso se a coluna 4 (LALR1) do cartão PROCED estiver preenchida e não houve erro grave na análise sintática da gramática de entrada.

1.2.10- Relatório da função GOTO(LR(1))

Contém a função GOTO (I,X) para cada conjunto de itens I, listando os pares (símbolos X da gramática para os quais esta função existe, os conjuntos de itens resultados desta função).

Este relatório só é impresso se a coluna 5 (LR1) do cartão PROCED estiver preenchida, e não houve erro grave na análise sintática da gramática de entrada.

1.2.11- Relatório de conjuntos de itens LR(1)

Contém para todos os conjuntos de itens LR(1) todos os itens LR(1) a este pertencentes. Um item LR(1) é representado por um terno (produção, posição ponto, terminal "lookahead").

Este relatório só é impresso se a coluna 5(LR1)do cartão PROCED estiver preenchida e não houve erro grave na análise sintática da gramática de entrada.

1.2.12- Relatório da Tabela de "parser" SLR(1)

Contém para cada estado que é identificado por um conjunto de itens, os ternos (terminais que contenham uma ação, ação correspondente, estado ou produção a esta ação relacionada). Para ações de "shift" a relação é para estado resultado de um deslocamento, para ações de "reduce" a relação é para produção pela qual ocorre a redução.

A ação correspondente a aceitação de uma sentença analisada sintaticamente por esta tabela corresponde a uma redução pela produção 0.

Este relatório só é impresso se a coluna 3(SLR) do cartão PROCED estiver preenchida e não houve erro grave na análise sintática da gramática de entrada.

1.2.13- Relatório da tabela de "parser" LALR(1)

Idêntico ao relatório da tabela de "parser" SLR (1), só que a coluna que indica a impressão do relatório é a 4 (LALR1) do cartão PROCED.

1.2.14- Relatório da tabela de "parser" LR(1)

Idêntico ao relatório da tabela de "parser" SLR (1), só que a coluna que indica a impressão do relatório é a 5 (LR1) do cartão PROCED.

1.2.15- Relatório de conflitos da tabela de "parser" SLR(1)

Contém para cada vez em que o algorítmo de geração da tabela de "parser" identificar um conflito; o estado ou conjunto de item, o terminal, a ação anterior e a ação nova.

Este relatório só é impresso se a coluna 3 (SLR) do cartão PROCED estiver preenchida, existir pelo menos um conflito para esta tabela e não houver erro grave na análise sintática da gramática de entrada.

1.2.16- Relatório de conflitos da tabela de"parser" LALR (1)

Idêntico ao relatório de conflitos da tabela de "parser" SLR(1), só que a coluna que indica a impressão do relatório é a 4 (LALR1).

1.2.17- Relatório de conflitos da tabela de "parser" LR (1)

Idêntico ao relatório de conflitos da tabela de "parser" SLR(1), só que a coluna que indica a impressão do relatório é a 5 (LR1).

1.2.18- Leiautes dos relatórios

Ver ANEXO II.

2. Mensagens de erro e ações para erros de índice inválido

2.1-Mensagem "TABELA DE TERMINAIS ESTA CHEIA"

2.1.1- Descrição

Esta mensagem é dada quando a área que contém os nomes dos terminais não tem mais espaço disponível. Este erro ocorre na conversão da gramática de entrada.

2.1.2- Ação do programa

O conversor não para ignorando deste ponto em diante todo terminal cujo nome não cabe no espaço ainda disponível.

Este erro é considerado um erro grave e impede a geração das tabelas dos analisadores sintáticos SLR(1), LALR(1) e LR(1).

2.1.3- Ação para correção

Aumentar o tamanho contido no cartão PARMCONV.

2.2- Mensagem "TABELA DE NAO TERMINAIS ESTA CHEIA"

Observações idênticas ao item anterior; só que relativos a área de armazenamento de nomes de não terminais.

2.3- Mensagem "CARACTER DESCONHECIDO α IGNORADO"

2.3.1-Descrição

É dada quando o analisador léxico do conversor da gramática de entrada identifica um caracter α cuja representação EBCDIC é desconhecida, deve-se observar que α só será impresso se a representação EBCDIC deste caracter tem correspondência no "FORMAT A" da linguagem ALGOL.

2.3.2-Ação do programa

Pula este caracter, e tudo se passa como se a coluna do cartão de entrada que o contém não existisse.

2.3.3- Ação para correção

Colocar o caracter correto.

2.4- Mensagem "IDENTIFICADOR AUSENTE"

2.4.1- Descrição

É dada quando o analisador sintático do conversor da gramática de entrada esperava um identificador como próximo símbolo para análise e aparece outro diferente deste.

2.4.2- Ação do programa

Insere um identificador antes do símbolo onde foi reconhecido o erro e prossegue a análise.

Este erro é considerado um erro grave e impede a geração das tabelas SLR(1), LALR(1) e LR(1).

2.4.3- Ação para correção

Verificar se houve ausência de identificador ou outro tipo de erro que possa provocar esta mensagem.

2.5- Mensagem "ESPAÇO AUSENTE"

2.5.1- Descrição

É dada quando o analisador sintático do conversor da gramática de entrada esperava um espaço como próximo símbolo para análise e aparece outro diferente deste.

2.5.2- Ação do programa

Insere um espaço antes do símbolo onde foi re-

conhecido o erro e prossegue a análise.

2.5.3- Ação para correção

Verificar se houve ausência de espaço ou outro tipo de erro que possa provocar esta mensagem.

2.6- Mensagem "=AUSENTE"

Observações idênticas ao item (V.2.5) só que para o símbolo =.

2.7- Mensagem "| AUSENTE"

Observações idênticas ao item (V.2.5) só que para o símbolo |.

2.8- Mensagem "; AUSENTE"

Observações idênticas ao item (V.2.5) só que para o símbolo ;.

2.9- Ações para problemas de índice inválido.

Quando o programa cancela por índice inválido, o procedimento a ser adotado deve ser o de reanalisar os cartões parâmetros utilizados. Se está se utilizando do valor assumido na inexistência do cartão, estes valores não comportam o tamanho da gramática e o cartão parâmetro envolvido no índice inválido passa a ser obrigatório. Se não, o valor indicado não é suficiente e deve ser recalculado e aumentado no cartão parâmetro envolvido.

3. Exemplos de utilização do gerador para a criação de tabelas SLR(1), LALR(1) e LR(1).

São apresentados diversos tipos de gramática cada uma com uma característica diferente. Em todos os exemplos utilizou- se dos valores "default", só sendo preenchidos os cartões do arquivo CARD que são listados no relatório GRAMÁTICAS DE ENTRADA E MENSAGENS DE ERRO, e o cartão PROCED com os cinco procedimentos FIRST, FOLLOW, SLR, LALR1, LR1 preenchidos.

Ver os relatórios produzidos pelo gerador no ANEXO II.

- Gramática com produção ε, SLR(1), LALR(1) e LR(1)

  E → TE'

  E' → + TE' | ε

  T → FT'

  T' → * FT'

  F → (E)| id

- Gramática LR(1), LALR(1) e não SLR(1)

  S → L = R |R

  L → * R| id

  R → L

- Gramática não SLR(1), não LALR(1) e não LR(1)

  E → E+E |E* E |(E) | id

- Gramática LR(1), não LALR(1) e não SLR(1)

  S → aAd |bBd |aBe |bAe

  A → c

  B → c

4. Exemplo de utilização das tabelas de análise sintática LR

O gerador implementado utiliza-se na etapa de conversão de gramática descrita anteriormente (V.1.1.1). Esta gramática é LALR(1) e utilizou-se deste gerador para criar as tabelas de análise sintática LALR(1).

Na documentação do conversor (IV.1) são explicados o relacionamento desta tabela de análise sintática com a análise léxica que passa os símbolos (espaço, =, | ; , cadeia, identificador, $) para o analisador sintático; com a recuperação de erro montada nas entradas em branco da tabela LALR(1); e com a geração simplificada de código que no caso é a representação interna intermediária da gramática de entrada do gerador, esta geração é montada em algumas entradas válidas na tabela LALR(1).

Os relatórios relativos a geração das tabelas de análise sintática LALR(1) acima, e também SLR(1) e LR(1) encontram- se no ANEXO III.

VI.CONCLUSÕES

1. Análise de desempenho

Foram realizados testes com gramáticas listadas no ANEXO IV e para cada uma destas levantou-se o tempo de CPU gasto para geração das tabelas SLR(1), LALR(1) e LR(1).

Os dados obtidos fazem parte da tabela abaixo:

				Tempo CPU (min.seg)			Nº Estados Tabela	
G	P	NT	T	SLR(1)	LALR(1)	LR(1)	SLR(1)/LALR(1)	LR(1)
1	3	2	2	.03	.04	.03	6	9
2	7	5	5	.04	.04	.05	15	25
3	11	7	6	.05	.06	.06	20	20
4	22	11	15	.08	.10	.20	45	152
5	35	17	20	.11	.17	.54	63	298
6	172	97	51	1.05	2.38	22.15	288	1399

G = Gramática P = Nº produções NT = Não Terminais

T = Terminais

O tempo levantado não pode ser utilizado para conclusões quanto a ordem do tempo gasto pelo algorítmo. Existem várias etapas de cálculo necessárias e diferentes, dependendo do tipo de tabela calculado e cada uma com ordem diferente. Também as gramáticas da tabela acima são amostras válidas como estimativa para comportamento de linguagens de programação usuais, porém podem ser viciadas para cálculo da ordem de tempo gasto nos algorítmos.

2. Sugestões para extensão do trabalho

2.1- Análise prévia da gramática de entrada

Existem alguns erros na codificação da gramática de entrada que podem ser identificadas antes da geração das tabelas de análise sintática SLR(1), LALR(1) e LR(1). Por exemplo, símbolos não terminais que não são alcançáveis a partir do símbolo inicial.

Outros tipos de erros podem ser pesquisados de forma a evitar o tempo desnecessário de geração das tabelas, e também o tempo maior para análise destes a partir de incoerências encontradas nas tabelas de análise sintática.

2.2- Recuperação de erro automática

As tabelas de análise sintática LR são muito úteis na análise da recuperação de erro.

Um esquema de recuperação de erro gerado automaticamente a partir das tabelas de análise sintática poderia ser pensado para gramáticas de tipo LR(1), pois estas identificam o erro no instante exato em que este ocorre. Porém o número elevado de estados da tabela tornam estas tabelas de pouca utilização prática. As gramáticas SLR(1) e LALR(1) possuem um número razoável de estados nas suas tabelas, mas o erro pode ser identificado em ponto adiante da sua ocorrência o que geraria complexidade adicional no estudo da recuperação automática do erro.

3. Observações Finais

Os dados em (VI.1) mostram que as tabelas de análise sintática LR(1) apresentam um tempo de cálculo e número de estados muito elevado. Pode ser considerado pouco prática a utilização de analisador sintático LR(1) para gramática com um número de produções maior do que 100, a menos que sejam usados algorítmos de compactação destes estados.

O comportamento do algorítmo é bastante satisfatório para cálculo das tabelas SLR(1) e LALR(1), completando o objetivo da Tese que é tornar disponível um gerador de tabelas de análise sintática para gramáticas LR, de utilização eficiente para o tamanho das gramáticas de linguagens de programação de aplicação prática, que fica em torno de 500 produções.

A N E X O I

```
BEGIN
  FILE
    PARMGRAM(KIND=READER);
  INTEGER
    NUMTSIMBPROD,
    NUMNACTERM;
 PROCEDURE PROCESSOCOMPLETO;
 BEGIN
  FILE
    IMPRE(KIND=PRINTER,FORMMESSAGE="2C1-8LP."),
    IMPRE1(KIND=PRINTER,FORMMESSAGE="2C1-8LP."),
    IMPRE2(KIND=PRINTER,FORMMESSAGE="2C1-8LP."),
    IMPRE3(KIND=PRINTER,FORMMESSAGE="2C1-8LP."),
    CARD(KIND=READER),
    PARMCONV(KIND=READER),
    PARMGERAL(KIND=READER),
    PROCED(KIND=READER),
    PARMLRC(KIND=READER),
    PARMLALR(KIND=READER),
    PARMLR1(KIND=READER);
  INTEGER
    I,
    J,
    AREAMAX,
    SHIFT,
    REDUCE,
    ACCEPT,
    TOTNACTERM,
    TOTTERM,
    LDIRDISP,
    CTAMPROD,
    CONTPR,
   INDMAX,
   NUMTERM,
   NUMPROD,
  SIZECITEM,
   SIZEITEMLALR1,
  SIZEITEM,
  SIZESTKCLOS,
    SIZEITEMLR1,
```

```
   SIZECITEMLR1,
 TAMMAXPROC;
 BOOLEAN
   HOUVEERROGRAVE;
 ALPHA
  ARRAY
   MATPROC1 1:5 ;
 INTEGER
  ARRAY
   MATPONTLDIR 1:NUMTSIMBPROD ,
   MATLDIRPROD 1:NUMTSIMBPROD ,
   MATLADOESQ 1:NUMNAOTERM ,
   MATPROC2 1:5,1:13 ,
   MATPROC3 1:13 ;
 DEFINE
   PROXLDIR(POS)=  MATPONTLDIR POS . 47:16 #,
   FIMPROC(POS) =  MATPONTLDIR POS . 31:1 #,
   INDTERM(POS) =  MATPONTLDIR POS . 30:1 #,
   PRODEPSILON(SIMB)=MATLADOESQ SIMB . 15:1 #,
   POSPRIMSIMBLDIR(SIMB)=MATLADOESQ SIMB . 47:16 #,
   PROC(X)=MATPROC3 X = 1#;
%*****************************************************************************
% ROTINA  CUJA FUNCAO E CONVERTER A GRAMATICA DE ENTRADA CODIFICADA NO*
% ARQUIVO CARD PARA REPRESENTACAO NA MEMORIA                          *
%*****************************************************************************
PROCEDURE CONVERSOR;
BEGIN
 INTEGER
   I,
    J,
   NUMNACTER ,
   NUMTER ,
   COLUNM,
   CLASSE,
   NCHAR,
   TOPO,
   LADOESQ,
   SIMB,
   SIMBOL;
 ALPHA
```

```
  CHAR;
 ALPHA
  ARRAY
   CARTAO 1:80 ,
   CHARARRAY 1:1 ,
   CHARS 1:ENTIER(AREAMAX/6)+1 ,
   MATNOMENAOTERM 1:ENTIER(AREAMAX/6)+1 ,
   MATNOMETERM 1:ENTIER(AREAMAX/6)+1 ;
 INTEGER
  ARRAY
   MATESTSEGUINTE 1:5,0:12 ,
   MATOPERACOES 1:5,0:12,1:4 ,
   MATACTION 0:20,1:7 ,
   MATACTIONOPER 0:20,1:7 ,
   MATACTIONCAUX 0:20,1:7 ,
   MATPROD 0:11 ,
   MATGOTO 0:20,1:7 ,
   STACKCITEM 1:21 ,
   STACKSIMB 1:21 ,
   MATCLASSE 0:255 ;
 BOOLEAN
   HOUVERECUPERRO;
 DEFINE
   TERRA=0#,
   OPERACAO(CITEM,SIMB)= MATACTIONOPER CITEM,SIMB #,
   ACAO    (CITEM,SIMB)= MATACTION CITEM,SIMB #,
   CODAUX  (CITEM,SIMB)= MATACTIONCAUX CITEM,SIMB #,
   CITEMTOPO=STACKCITEM TOPO #,
   GOTO(CITEM,NTER)= MATGOTO CITEM,NTER #,
   TAMPROD(PROD)   =MATPROD PROD . 47:16 #,
    POSULTSIMBLDIR(SIMB)=MATLADOESQ SIMB . 31:16 #,
   NAOTERMESQ(PROD)=MATPROD PROD . 31:16 #;
%*****************************************************************
% ROTINA QUE OBTEM O PROXIMO CARACTER . SE O PROXIMO CARACTER     *
% PERTENCE AO PROXIMO CARTAO LE ESTE CARTAO                       *
% VARIAVEIS SAIDA:  CHAR  - CONTEM CARACTER LIDO                  *
%                   CLASSE - CONTEM CLASSE DO CARACTER            *
% OBS: SE TERMINOU O ARQUIVO CARD MOVE VALOR 12 PARA CLASSE       *
%*****************************************************************
PROCEDURE GETCHAR;
```

```
BEGIN
 INTEGER
   I;
 IF COLUNM >= 81
  THEN
    BEGIN
    WRITE(IMPRE SPACE 1 ,<80A1>,FOR I:=1 STEP 1 UNTIL 80 DO CARTAO I );
    IF READ(CARD,<80A1>,FOR I:= 1 STEP 1 UNTIL 80 DO CARTAO I )
     THEN
      CLASSE:= 12
     ELSE
      BEGIN
      COLUNM:= 1;
      CHAR:= CARTAO COLUNM ;
      CLASSE:= MATCLASSE CHAR. 7:8  ;
      COLUNM:= COLUNM + 1;
      END
    END
  ELSE
   BEGIN
    CHAR:= CARTAO COLUNM ;
    CLASSE:= MATCLASSE CHAR. 7:8  ;
    COLUNM:= COLUNM + 1;
   END;
END GETCHAR;
%                          ****************************
%*************************************************************************
% ROTINA CHAMADA PELA ROTINA CONVERSOR SEMPRE QUE NECESSARIO OBTER O  *
% PROXIMO SIMBOLO DA SENTENCA DE ENTRADA                              *
%*************************************************************************
PROCEDURE SCANNER;
BEGIN
 POINTER
   PC,
   PT,
   PNT;
 BOOLEAN
    ENCONTROU;
 INTEGER
    ESTADOATUAL,
```

```
    CLASSEATUAL,
    I;
PC:= POINTER(CHARS);
NCHAR:= 0;
 ESTADOATUAL:= 1;
 WHILE ESTADOATUAL ¬= 6 DO
  BEGIN
   CLASSEATUAL:= CLASSE;
   FOR I:= 1 STEP 1 WHILE MATOPERACOES ESTADOATUAL,CLASSEATUAL,I ¬= 0
                                        AND I <= 3 DO
     CASE MATOPERACOES ESTADOATUAL,CLASSEATUAL,I  - 1  OF
      BEGIN
       %  1  ADD
       IF NCHAR < AREAMAX
        THEN
         BEGIN
          CHARARRAY 1 := CHAR;
          REPLACE PC:PC BY POINTER(CHARARRAY)+5 FOR 1;
          NCHAR:= NCHAR + 1;
         END;
       %  2  GETCHAR
       GETCHAR;
       %  3  GETFIRSTCHARCARTAO
       BEGIN
        GETCHAR;
        WHILE COLUNM ¬= 2 DO
         GETCHAR;
       END;
       %  4  ADDPLIC
         BEGIN
          PC:= PC - 1;
          REPLACE PC:PC BY "'";
         END;
       %  5  LOOKUPTERM
       BEGIN
        PT:= POINTER(MATNOMETERM);
        PC:= POINTER(CHARS);
        NUMTER := 0;
        ENCONTROU:= FALSE;
        WHILE ¬ENCONTROU AND NUMTER  ¬= TOTTERM   DO
```

```
  BEGIN
   NUMTER := NUMTER  + 1;
   IF PC = PT FOR NCHAR AND PT+NCHAR = 4"01" FOR 1
    THEN ENCONTROU:= TRUE
    ELSE
     BEGIN
      SCAN PT:PT UNTIL = 4"01" ;
      PT:= PT + 1;
     END;
  END;
 IF ¬ENCONTROU
  THEN
   IF DELTA(PT,POINTER(MATNOMETERM))+NCHAR > AREAMAX
    THEN
     BEGIN
      WRITE(IMPRE SPACE 1 ,<"TABELA DE TERMINAIS CHEIA">);
      HOUVEERROGRAVE := TRUE;
     END
    ELSE
     BEGIN
       NUMTER :=TOTTERM:= TOTTERM + 1;
      REPLACE PT:PT BY PC FOR NCHAR;
      REPLACE PT BY 4"01" FOR 1;
     END;
END;
%  6  LOOKUPNAOTERM
BEGIN
 PNT:= POINTER(MATNOMENACTERM);
 PC:= POINTER(CHARS);
 NUMNAOTER := 0;
 ENCONTROU:= FALSE;
 WHILE ¬ENCONTROU AND NUMNAOTER  ¬= TOTNAOTERM   DO
  BEGIN
   NUMNAOTER := NUMNAOTER  + 1;
   IF PC = PNT FOR NCHAR AND PNT+NCHAR = 4"01" FOR 1
    THEN ENCONTROU:= TRUE
    ELSE
     BEGIN
      SCAN PNT:PNT UNTIL = 4"01" ;
      PNT:= PNT + 1;
```

```
            END;
          END;
         IF ¬ENCONTROU
          THEN
           IF DELTA(PNT,POINTER(MATNOMENACTERM))+NCHAR > AREAMAX
            THEN
             BEGIN
              WRITE(IMPRE SPACE 1 ,<"TABELA DE NAO TERMINAIS CHEIA">);
              HOUVEERROGRAVE := TRUE;
             END
            ELSE
             BEGIN
              NUMNAOTER :=TOTNAOTERM := TOTNACTERM + 1;
              REPLACE PNT:PNT BY PC FOR NCHAR;
              REPLACE PNT BY 4"01" FOR 1;
             END;
        END;
        SIMBOL:= 2;       %  7
        SIMBOL:= 4;       %  8
        SIMBOL:= 3;       %  9
        SIMBOL:= 1;       %  10
        SIMBOL:= 6;       %  11
        SIMBOL:= 5;       %  12
        SIMBOL:= 7;       %  13
        WRITE(IMPRE SPACE 1 ,<"CARACTER DESCONHECIDO ",A1,
                              " IGNORADO">,CHAR);   %  14
         % 15
         BEGIN
          PC:= PC - 1;
          NCHAR:= NCHAR - 1;
         END;
       END;
    ESTADOATUAL:= MATESTSEGUINTE ESTADOATUAL,CLASSEATUAL ;
  END;
END SCANNER;
%                              ******************************
%*****************************************************************************
% ROTINA QUE EXECUTA AS OPERACOES RELATIVAS A MONTAGEM DA                     *
% REPRESENTACAO NA MEMORIA DA GRAMATICA DE ENTRADA                            *
%*****************************************************************************
```

```
PROCEDURE REALIZAOPERACAO(SIMBOL);
INTEGER SIMBOL;
BEGIN
 CASE OPERACAO(CITEMTOPO,SIMBOL)-1 OF
  BEGIN
   BEGIN
    LADOESQ:= NUMNAOTER;
   CTAMPROD:= 0;
   END;
   BEGIN
    IF POSPRIMSIMBLDIR(LADOESQ)= TERRA
     THEN
      POSPRIMSIMBLDIR(LADOESQ):= LDIRDISP
     ELSE
      PROXLDIR(POSULTSIMBLDIR(LADOESQ)):= LDIRDISP;
    MATLDIRPROD LDIRDISP := NUMNAOTER;
    POSULTSIMBLDIR(LADOESQ):= LDIRDISP;
    PROXLDIR(LDIRDISP):= TERRA;
    LDIRDISP:= LDIRDISP + 1;
    CTAMPROD:= CTAMPROD + 1;
   END;
   BEGIN
    IF POSPRIMSIMBLDIR(LADOESQ) = TERRA
     THEN
      POSPRIMSIMBLDIR(LADOESQ):= LDIRDISP
     ELSE
      PROXLDIR(POSULTSIMBLDIR(LADOESQ)):= LDIRDISP;
    MATLDIRPROD LDIRDISP := NUMTER;
    POSULTSIMBLDIR(LADOESQ):= LDIRDISP;
    INDTERM(LDIRDISP):= 1;
    PROXLDIR(LDIRDISP):= TERRA;
    LDIRDISP:= LDIRDISP + 1;
    CTAMPROD:= CTAMPROD + 1;
   END;
   BEGIN
    IF POSPRIMSIMBLDIR(LADOESQ) ¬= TERRA
     THEN
      BEGIN
    FIMPROD(POSULTSIMBLDIR(LADOESQ)):= 1;
    CONTPR:= CONTPR + 1;
```

```
    IF CTAMPROD > TAMMAXPROD
     THEN TAMMAXPROD:= CTAMPROD;
      END;
   END;
  BEGIN
   PRODEPSILON(LADOESQ):= 1;
   CONTPR:= CONTPR + 1;
  END;
  END;
END REALIZACPERACAO;
%                              ******************************
%*****************************************************************************
% ACAO EMPILHA DE ANALISE SINTATICA LR                                       *
%*****************************************************************************
PROCEDURE SHIFT(SIMBOL);
INTEGER SIMBOL;
BEGIN
 INTEGER
  CITEMANT;
 STACKSIMB TOPO := SIMBOL;
 CITEMANT:= CITEMTOPO;
 TOPO:= TOPO + 1;
 CITEMTOPO:= CODAUX(CITEMANT,SIMBOL);
END SHIFT;
%                              ******************************
%*****************************************************************************
% ACAO REDUZA DE ANALISE SINTATICA LR                                        *
%*****************************************************************************
PROCEDURE REDUCE(SIMBOL);
INTEGER SIMBOL;
BEGIN
 INTEGER
   CITEMANT,
   PRODUCAO;
 PRODUCAO:= CODAUX(CITEMTOPO,SIMBOL);
 TOPO:= TOPO - TAMPROD(PRODUCAO);
 STACKSIMB TOPO := NAOTERMESQ(PRODUCAO);
 CITEMANT:= CITEMTOPO;
 TOPO:= TOPO + 1;
 CITEMTOPO:= GOTO(CITEMANT,STACKSIMB TOPO-1 );
```

```
END REDUCE;
%                              ****************************
%*************************************************************************
% ROTINAS DE RECUPERACAO DE ERRO COM INSERCAO DE SIMBOLO ESPERADO E      *
% IMPRESSAO DE MENSAGEM DE ERRO RESPECTIVA                               *
%*************************************************************************
PROCEDURE INSERE(SIMBOL);
INTEGER SIMBOL;
BEGIN
 SIMB:= SIMBOL;
END INSERE;
PROCEDURE RECUPERRO;
BEGIN
 HOUVERECUPERRO:= TRUE;
 CASE CODAUX(CITEMTOPO,SIMBOL)   OF
  BEGIN
   BEGIN
    WRITE(IMPRE SPACE 1 ,<"ERRO DE PROGRAMA POSICAO NAO ALCANCAVEL ">);
   END;
   BEGIN
    WRITE(IMPRE SPACE 1 ,<"ESPACO AUSENTE">);
    INSERE(1);
   END;
    BEGIN
    WRITE(IMPRE SPACE 1 ,<"IDENTIFICADOR AUSENTE">);
    INSERE(5);
    HOUVEERROGRAVE:= TRUE;
   END;
   BEGIN
    WRITE(IMPRE SPACE 1 ,<"= AUSENTE">);
    INSERE(2);
   END;
   BEGIN
    WRITE(IMPRE SPACE 1 ,<"| AUSENTE">);
    INSERE(4);
   END;
   BEGIN
    WRITE(IMPRE SPACE 1 ,<"; AUSENTE">);
    INSERE(3);
   END;
```

```
  END;
END RECUPERPRO;
%                              ****************************
%*************************************************************************
% ROTINA DE IMPRESSAO DE RELATORIO COM REPRESENTACAO INTERNA E NOME    *
% DOS NAC TERMINAIS                                                     *
%*************************************************************************
PROCEDURE IMPNAOTERMINAIS;
BEGIN
 INTEGER
   CONTLINHA,
   I,
   J;
 ALPHA
  ARRAY
   NOME 1:20 ;
 POINTER
   PA,
   PN,
   PT;
PROCEDURE CABECALHO;
BEGIN
 WRITE(IMPRE1 SKIP 1 );
 WRITE(IMPRE1,<X42,"NAOTERMINAIS">);
 WRITE(IMPRE1 SPACE 1 ,<X36,"REPRESENTACAO INTERNA">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 WRITE(IMPRE1 SPACE 1 ,<"  NAO    |">);
 WRITE(IMPRE1 SPACE 1 ,<"TERMINAL|",X18,"NOME">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 CONTLINHA:= 0;
END CABECALHO;
 CABECALHO;
 PT:= POINTER(MATNOMENAOTERM);
 WRITE(IMPRE1 SPACE 1 );
 WRITE(IMPRE1,<X6,"0",X3,"S'">);
 CONTLINHA:=.CONTLINHA + 2;
 FOR I:= 1 STEP 1 UNTIL TOTNAOTERM DO
  BEGIN
   PN:= POINTER(NOME);
   PA:= PT;
```

```
   REPLACE PN BY " " FOR 120;
   WHILE PT¬= 4"01" FOR 1 AND DELTA(PT,PA)¬=120 DO
     REPLACE PN:PN BY PT:PT FOR 1;
   IF PT¬= 4"01" FOR 1
    THEN
     BEGIN
      SCAN PT:PT UNTIL = 4"01";
     END;
   PT:= PT + 1;
   IF CONTLINHA > 75
    THEN CABECALHO;
   WRITE(IMPRE1 SPACE 1 );
   WRITE(IMPRE1          ,<X1,I6,X3,20A6>,I,
                FOR J:= 1 STEP 1 UNTIL 20 DO NOME J );
   CONTLINHA:= CONTLINHA + 2;
  END;
   IF CONTLINHA > 75
    THEN CABECALHO;
END IMPNACTERMINAIS;
%                              ******************************
%*************************************************************************
% ROTINA DE IMPRESSAO DE RELATORIO COM REPRESENTACAO INTERNA E NOME    *
% DOS TERMINAIS                                                        *
%*************************************************************************
PROCEDURE IMPTERMINAIS;
BEGIN
 INTEGER
   CONTLINHA,
   I,
   J;
 ALPHA
  ARRAY
   NOME 1:20 %;
 POINTER
   PA,
   PN,
   PT;
PROCEDURE CABECALHO;
BEGIN
 WRITE(IMPRE1 SKIP 1 );
```

```
 WRITE(IMPRE1,<X42,"TERMINAIS">);
 WRITE(IMPRE1 SPACE 1 ,<X36,"REPRESENTACAO INTERNA">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 WRITE(IMPRE1 SPACE 1 ,<"TERMINAL",X18,"NOME">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 CONTLINHA:= 0;
END CABECALHO;
 CABECALHO;
 PT:= POINTER(MATNOMETERM);
 FOR I:= 1 STEP 1 UNTIL TOTTERM DO
  BEGIN
   PN:= POINTER(NOME);
   PA:= PT;
   REPLACE PN BY " " FOR 120;
   WHILE PT¬= 4"01" FOR 1 AND DELTA(PT,PA)¬=120 DO
     REPLACE PN:PN BY PT:PT FOR 1;
   IF PT¬= 4"01" FOR 1
    THEN
     BEGIN
      SCAN PT:PT UNTIL = 4"01";
     END;
   PT:= PT + 1;
   IF CONTLINHA > 75
    THEN CABECALHO;
   WRITE(IMPRE1 SPACE 1 );
   WRITE(IMPRE1          ,<X1,I6,X3,20A6>,I+TOTNAOTERM,
                FOR J:= 1 STEP 1 UNTIL 20 DO NOME J );
   CONTLINHA:= CONTLINHA + 2;
  END;
   IF CONTLINHA > 75
    THEN CABECALHO;
 WRITE(IMPRE1 SPACE 1 );
 WRITE(IMPRE1,<X1,I6,X3,"EOF ($)">,TOTTERM+TOTNAOTERM+1);
END IMPTERMINAIS;
%                          ****************************
 WRITE(IMPRE SPACE 1 ,<132("-")>);
 WRITE(IMPRE SPACE 1 ,<X20,"GRAMATICA DE ENTRADA E MENSAGENS DE ERRO">)
                                                                    ;
 WRITE(IMPRE SPACE 1 ,<132("-")>);
%********   INICIALIZACAO DAS TABELAS DO CONVERSOR
```

```
FOR I:= 1,2,3,12                          DO MATESTSEGUINTE 1,I := 6;
FOR I:= 4,5                               DO MATESTSEGUINTE 1,I := 4;
FOR I:= 0,6,8,9,11                        DO MATESTSEGUINTE 1,I := 1;
FOR I:= 0,1,2,3,4,5,6,7,9,11              DO MATESTSEGUINTE 2,I := 2;
FOR I:= 10,12                             DO MATESTSEGUINTE 2,I := 6;
FOR I:= 0 STEP 1 UNTIL 11                 DO MATESTSEGUINTE 3,I := 2;
FOR I:= 4,5,6,9                           DO MATESTSEGUINTE 4,I := 4;
FOR I:= 0,1,2,3,7,8,10,11,12              DO MATESTSEGUINTE 4,I := 6;
FOR I:= 0,1,2,3,4,5,6,9,10,11,12          DO MATESTSEGUINTE 5,I := 6;
FOR I:= 7,8                               DO MATESTSEGUINTE 5,I := 5;
FOR I:= 4,5                               DO MATOPERACOES 1,I,1 := 1;
FOR I:= 0,4,5                             DO MATOPERACOES 1,I,2 := 2;
FOR I:= 1,2,3,6,7,8,9,10,11               DO MATOPERACOES 1,I,1 := 2;
FOR I:= 1,2,3,4,5,6,7,8,9,11              DO MATOPERACOES 2,I,1 := 1;
FOR I:= 0,1,2,3,4,5,6,7,8,9,11            DO MATOPERACOES 2,I,2 := 2;
FOR I:= 0,5                               DO MATOPERACOES 3,I,2 := 2;
FOR I:= 1,2,3,4,6,7,8,9,10,11             DO MATOPERACOES 3,I,1 := 2;
FOR I:= 4,5,6,9                           DO MATOPERACOES 4,I,1 := 1;
FOR I:= 4,5,6,9                           DO MATOPERACOES 4,I,2 := 2;
FOR I:= 0,1,2,3,7,8,10,11,12              DO MATOPERACOES 4,I,1 := 6;
FOR I:= 0,1,2,3,7,8,10,11,12              DO MATOPERACOES 4,I,2 :=12;
FOR I:= 0,1,2,3,4,5,6,9,10,11,12          DO MATOPERACCES 5,I,1 :=10;
MATESTSEGUINTE 1,10 := 2;
MATESTSEGUINTE 1,07 := 5;
MATESTSEGUINTE 2,08 := 3;
MATESTSEGUINTE 3,12 := 6;
MATOPERACCES 2, 0,1 := 2;
MATOPERACCES 2,10,3 :=11;
MATOPERACCES 1, 1,2 := 7;
MATOPERACCES 1, 2,2 := 8;
MATOPERACOES 1, 3,2 := 9;
MATOPERACOES 3, 5,1 := 4;
MATOPERACCES 2,10,1 := 5;
MATOPERACCES 2, 0,1 := 14;
MATOPERACCES 1, 0,1 := 14;
MATOPERACCES 2,10,2 := 2;
MATOPERACCES 1,12,1 := 13;
MATOPERACCES 2,12,1 := 13;
MATOPERACCES 3,12,1 := 13;
MATOPERACCES 5, 7,1 := 2;
```

```
MATOPERACCES 5, 8,1 := 3;
MATOPERACCES 1, 8,1 := 3;
MATOPERACCES 3, 0,1 := 15;
FOR I:= 193 STEP 1 UNTIL 201          DO MATCLASSE I := 4;
FOR I:= 209 STEP 1 UNTIL 215          DO MATCLASSE I := 4;
MATCLASSE 216 := 5;
MATCLASSE 217 := 4;
FOR I:= 226 STEP 1 UNTIL 233          DO MATCLASSE I := 4;
FOR I:= 240 STEP 1 UNTIL 249          DO MATCLASSE I := 6;
MATCLASSE 126 :=  1;
MATCLASSE  79 :=  2;
MATCLASSE  94 :=  3;
MATCLASSE  64 :=  7;
MATCLASSE 123 :=  8;
MATCLASSE 109 :=  9;
MATCLASSE 125 := 10;
FOR I:=74,75,76,77,78,80,90,91,92,93,95,96,97,107,108,110,111,122,124,
          127   DO MATCLASSE I := 11;
FOR I:= 0 STEP 1 UNTIL 20 DO
ACAC( 0,1):= 1;
ACAC( 1,5):= 1;
ACAC( 1,7):= 3;
ACAC( 2,5):= 2;
ACAC( 2,7):= 2;
ACAC( 3,1):= 1;
ACAC( 4,2):= 1;
ACAC( 5,1):= 1;
ACAC( 6,1):= 2;
ACAC( 7,5):= 2;
ACAC( 7,7):= 2;
ACAC( 8,1):= 1;
ACAC( 9,2):= 2;
ACAC(10,3):= 1;
ACAC(10,4):= 1;
ACAC(11,3):= 2;
ACAC(11,4):= 2;
ACAC(11,5):= 1;
ACAC(11,6):= 1;
ACAC(12,3):= 2;
ACAC(12,4):= 2;
```

```
ACAC(12,5):= 2;
ACAC(12,6):= 2;
ACAC(13,1):= 2;
ACAC(14,1):= 1;
ACAC(15,1):= 1;
ACAC(16,1):= 1;
ACAC(17,1):= 2;
ACAC(18,3):= 2;
ACAC(18,4):= 2;
ACAC(18,5):= 1;
ACAC(18,6):= 1;
ACAC(19,3):= 2;
ACAC(19,4):= 2;
ACAC(19,5):= 2;
ACAC(19,6):= 2;
ACAC(20,3):= 2;
ACAC(20,4):= 2;
ACAC(20,5):= 2;
ACAC(20,6):= 2;
J:= 0; FOR I:= 2,1,1,1,1,1,1   DO BEGIN J:=J+1; CODAUX( 0,J):=I; END;
J:= 0; FOR I:=2,2,2,2,6,2,0    DO BEGIN J:=J+1; CODAUX( 1,J):=I; END;
J:= 0; FOR I:=2,2,2,2,1,2,1    DO BEGIN J:=J+1; CODAUX( 2,J):=I; END;
J:= 0; FOR I:=7,1,1,1,1,1,1    DO BEGIN J:=J+1; CODAUX( 3,J):=I; END;
J:= 0; FOR I:=3,8,3,3,3,3,3    DO BEGIN J:=J+1; CODAUX( 4,J):=I; END;
J:= 0; FOR I:=9,1,1,1,1,1,1    DO BEGIN J:=J+1; CODAUX( 5,J):=I; END;
J:= 0; FOR I:=10,1,1,1,1,1,1   DO BEGIN J:=J+1; CODAUX( 6,J):=I; END;
J:= 0; FOR I:=2,2,2,2,2,2,2    DO BEGIN J:=J+1; CODAUX( 7,J):=I; END;
J:= 0; FOR I:=12,1,1,1,1,1,1   DO BEGIN J:=J+1; CODAUX( 8,J):=I; END;
J:= 0; FOR I:=3,6,3,3,3,3,3    DO BEGIN J:=J+1; CODAUX( 9,J):=I; END;
J:= 0; FOR I:=4,5,13,14,0,0,5  DO BEGIN J:=J+1; CODAUX(10,J):=I; END;
J:= 0; FOR I:=2,2,4,4,6,17,2   DO BEGIN J:=J+1; CODAUX(11,J):=I; END;
J:= 0; FOR I:=5,5,7,7,7,7,5    DO BEGIN J:=J+1; CODAUX(12,J):=I; END;
J:= 0; FOR I:=3,1,1,1,1,1,1    DO BEGIN J:=J+1; CODAUX(13,J):=I; END;
J:= 0; FOR I:=12,1,1,1,1,1,1   DO BEGIN J:=J+1; CODAUX(14,J):=I; END;
J:= 0; FOR I:=19,1,1,1,1,1,1   DO BEGIN J:=J+1; CODAUX(15,J):=I; END;
J:= 0; FOR I:=20,1,1,1,1,1,1   DO BEGIN J:=J+1; CODAUX(16,J):=I; END;
J:= 0; FOR I:=11,1,1,1,1,1,1   DO BEGIN J:=J+1; CODAUX(17,J):=I; END;
J:= 0; FOR I:=2,2,5,5,6,17,2   DO BEGIN J:=J+1; CODAUX(18,J):=I; END;
J:= 0; FOR I:=5,5,8,8,8,8,5    DO BEGIN J:=J+1; CODAUX(19,J):=I; END;
J:= 0; FOR I:=5,5,9,9,9,9,5    DO BEGIN J:=J+1; CODAUX(20,J):=I; END;
```

```
OPERACAO( 1,5):= 1;
OPERACAO(10,3):= 4;
OPERACAO(10,4):= 4;
OPERACAO(11,5):= 2;
OPERACAO(11,6):= 3;
OPERACAO(12,3):= 5;
OPERACAO(12,4):= 5;
OPERACAO(18,5):= 2;
OPERACAO(18,6):= 3;
NAOTERMESQ( 0):= 0;
NAOTERMESQ( 1):= 1;
NAOTERMESQ( 2):= 1;
NAOTERMESQ( 3):= 2;
NAOTERMESQ( 4):= 3;
NAOTERMESQ( 5):= 3;
NAOTERMESQ( 6):= 4;
NAOTERMESQ( 7):= 5;
NAOTERMESQ( 8):= 5;
NAOTERMESQ( 9):= 5;
NAOTERMESQ(10):= 6;
NAOTERMESQ(11):= 7;
TAMPROD( 0):= 1;
TAMPROD( 1):= 1;
TAMPROD( 2):= 3;
TAMPROD( 3):= 4;
TAMPROD( 4):= 1;
TAMPROD( 5):= 3;
TAMPROD( 6):= 2;
TAMPROD( 7):= 1;
TAMPROD( 8):= 3;
TAMPROD( 9):= 3;
TAMPROD(10):= 1;
TAMPROD(11):= 1;
GOTO( 0,1):= 1;
GOTO( 1,2):= 3;
GOTO( 1,4):= 4;
GOTO( 1,6):= 5;
GOTO( 8,3):=10;
GOTO( 8,5):=11;
GOTO(11,6):=16;
```

```
 GOTO(11,7):=15;
 GOTO(14,5):=18;
 GOTO(18,6):=16;
 GOTO(18,7):=15;
%********
%********  ANALISE SINTATICA POR TABELA LR
 READ(CARD,<80A1>,FOR I:=1 STEP 1 UNTIL 80 DO CARTAO I );
 COLUNM:=1;
 GETCHAR;
 LDIRDISP:= 1;
 HOUVERECUPERRO:= FALSE;
 HOUVEERROGRAVE:= FALSE;
 TOPO:= 1;
 CITEMTOPO:= 0;
 SCANNER;
 SIMB:= SIMBOL;
 WHILE ACAO(CITEMTOPO,SIMB  ) ¬= ACCEPT DO
  BEGIN
   CASE ACAO(CITEMTOPO,SIMB  ) OF
     BEGIN
     RECUPERRO;
     BEGIN
      IF OPERACAO(CITEMTOPO,SIMB  )¬= 0
       THEN REALIZAOPERACAO(SIMB  );
      SHIFT(SIMB  );
      IF ¬HOUVERECUPERRO
       THEN
        BEGIN
         SCANNER;
         SIMB:= SIMBOL;
        END
       ELSE
        BEGIN
         SIMB:= SIMBOL;
         HOUVERECUPERRO:= FALSE;
        END;
     END;
     BEGIN
      IF OPERACAO(CITEMTOPO,SIMB  )¬= 0
       THEN REALIZAOPERACAO(SIMB  );
```

```
      REDUCE(SIMB  );
     END;
    END;
  END;
%********
%******** IMPRESSAO DOS TERMINAIS E NAC TERMINAIS
 IMPNAOTERMINAIS;
 IMPTERMINAIS;
%********
END CONVERSOR;
%                              ******************************
%*************************************************************************
% ROTINA QUE CONVERTE A GRAMATICA DA REPRESENTACAO SAIDA DO CONVERSOR *
% PARA REPRESENTACAO UTILIZADA PELO GERADOR DE TABELAS, E QUE TAMBEM  *
% CHAMA PROCEDIMENTOS (CARTAO PROCED) DESEJADOS                       *
%*************************************************************************
PROCEDURE GERPARSER;
BEGIN
 INTEGER ARRAY
   MATSIMBOL 0:NUMNAOTERM ,
   MATPRODUCAO 0:NUMTPROD ,
   MATLADODIREITOPROD 1:NUMTSIMBPROD+1 ,
   STACKCLOS 1:SIZESTKCLOS ,
    STACKCLOSPONT 1:SIZESTKCLOS ;
 INTEGER
   POS,
   TPROD,
   NPROD,
   CONTPROD,
   CONTLDIR,
   I,
   SE,
   PRIMITEMCLOS,
   NUMITEMCLOS;
 ALPHA ARRAY
   NAOT 1:3 ;
 BOOLEAN
   ARRAY
   MATGEREPSILON 0:ENTIER((NUMNAOTERM-1)/48) ,
   MATFOLLOW 0:NUMNAOTERM,0:INDMAX ,
```

```
  MATFIRST 1:NUMNAOTERM,0:INDMAX ;
BOOLEAN
  PRIMEIRAENTRADA;
DEFINE
  PROXITEMCLOS(ITEM)=STACKCLOSPONT ITEM . 47:24 #,
  ITEMCLOSANT(ITEM)=STACKCLOSPONT ITEM . 23:24 #,
  TERRA=0#,
  PRIMEIRONAOTERMINAL=1#,
  ULTIMONAOTERMINAL=NUMNAOTERM#,
  PRIMEIROTERMINAL=NUMNAOTERM+1#,
  ULTIMOTERMINAL=NUMNAOTERM+NUMTERM#,
  PRIMEIRAPRODUCAO(NTERM)=MATSIMBOL NTERM . 47:16 #,
  ULTIMAPRODUCAO(NTERM)=MATSIMBOL NTERM . 47:16 +
                        MATSIMBOL NTERM . 31:16 -1#,
  PRODUCAOEPSILON(PROD)=MATPRODUCAO PROD . 31:16  = 0#,
  POSPRIMSIMBOL(PROD)=MATPRODUCAO PROD . 47:16 #,
  POSULTSIMBOL(PROD)=MATPRODUCAO PROD . 47:16  +
                     MATPRODUCAO PROD . 31:16 -1 #,
  SIMBPROD(PSIMB)=MATLADODIREITOPROD PSIMB #,
  ETERMINAL=> NUMNAOTERM#,
  EPSILON=NUMNAOTERM+NUMTERM+1#,
  GERAEPSILON(SIMB)=MATGEREPSILON ENTIER((SIMB-1)/48) .
                     (SIMB-1 - 48*ENTIER((SIMB-1)/48)):1 #,
  DOLLAR=NUMNAOTERM+NUMTERM#,
  ENAOTERMINAL=<=NUMNAOTERM#,
  FOLLOW(SIMB,TERM)=MATFOLLOW SIMB,ENTIER((TERM-NUMNAOTERM-1)/48) .
                     (TERM-NUMNAOTERM-1 - 48*ENTIER((TERM-NUMNAOTERM
                                                    -1)/48)):1 #,
  PRODITEMCLOS(NITEM)=STACKCLOS NITEM . 47:16 #,
  POSPONTOITEMCLOS(NITEM)=STACKCLOS NITEM . 31:16 #,
  LOOKAHEADITEMCLOS(NITEM)=STACKCLOS NITEM . 15:16 #,
  POSSIMBPONTOITEMCLOS(NITEM)=MATPRODUCAO
                                PRODITEMCLOS(NITEM) . 47:16  +
                                POSPONTOITEMCLOS(NITEM)#,
  TAMPRODUCAO(PROD)=MATPRODUCAO PROD . 31:16 #,
  NAOTERMESQUERDO(PROD)=MATPRODUCAO PROD . 15:16 #,
  PRIMEIROSIMBOLO=1#,
  ULTIMOSIMBOLO=NUMNAOTERM+NUMTERM#,
  NACVAZIO=> 0#,
  FIRST(SIMB,TERM)=MATFIRST SIMB,ENTIER((TERM-NUMNAOTERM-1)/48) .
```

```
                    (TERM-NUMNAOTERM-1 - 48*ENTIER((TERM-
                                          NUMNAOTERM-1)/48)):1 #;
INTEGER PROCEDURE RESTO(DIVIDENDO,DIVISOR);
INTEGER DIVIDENDO,DIVISOR;
BEGIN
 RESTO:= DIVIDENDO - DIVISOR*ENTIER(DIVIDENDO/DIVISOR);
END RESTO;
%*************************************************************************
% ROTINA DE CALCULO DA RELACAO FIRST  PARA OS SIMBOLOS NAO TERMINAIS    *
% DA GRAMATICA DE ENTRADA                                               *
%*************************************************************************
PROCEDURE GERAFIRST;
BEGIN
 INTEGER
   NAOTERMINAL,
   PRODUCAO,
   POSSIMBOL,
   SIMBOL;
 BOOLEAN
   ADD,
   TERMINOU;
PROCEDURE ADDFIRST(SIMBOL,TERM);
INTEGER
  SIMBOL,
  TERM;
BEGIN
    IF ¬FIRST(SIMBOL,TERM)
     THEN
      BEGIN
       FIRST(SIMBOL,TERM):= TRUE;
       ADD:= TRUE;
      END;
END ADDFIRST;
PROCEDURE ADDFIRSTEPSILON(SIMBOL);
INTEGER SIMBOL;
BEGIN
    IF ¬GERAEPSILON(SIMBOL)
     THEN
      BEGIN
       GERAEPSILON(SIMBOL):= TRUE;
```

```
        ADD:= TRUE;
       END;
END ADDFIRSTEPSILON;
PROCEDURE ADDLINHAFIRST(NAOTERM1,NAOTERM2);
INTEGER
  NAOTERM1,
  NAOTERM2;
BEGIN
 INTEGER
   INDAUX;
 REAL
   VARAUX;
 FOR INDAUX:= 0 STEP 1 UNTIL INDMAX DO
  BEGIN
   VARAUX:= REAL(MATFIRST NAOTERM2,INDAUX );
   MATFIRST NAOTERM2,INDAUX := MATFIRST NAOTERM2,INDAUX  OR
                               MATFIRST NAOTERM1,INDAUX ;
   IF VARAUX. 47:48  ¬= REAL(MATFIRST NAOTERM2,INDAUX )
    THEN ADD:= TRUE;
  END;
END ADDLINHAFIRST;
 ADD:= TRUE;
 WHILE ADD DO
  BEGIN
   ADD:= FALSE;
   FOR NAOTERMINAL:= PRIMEIRONAOTERMINAL STEP 1 UNTIL
                     ULTIMONAOTERMINAL DO
    BEGIN
     FOR PRODUCAO:= PRIMEIRAPRODUCAO(NAOTERMINAL) STEP 1 UNTIL
                      ULTIMAPRODUCAO(NAOTERMINAL) DO
      BEGIN
       IF PRODUCAOEPSILON(PRODUCAO)
        THEN
         BEGIN
          ADDFIRSTEPSILON(NAOTERMINAL);
         END
        ELSE
         BEGIN
          TERMINOU:= FALSE;
          FOR POSSIMBOL:= POSPRIMSIMBOL(PRODUCAO) STEP 1 WHILE
```

```
                    POSSIMBOL <= POSULTSIMBOL(PRODUCAO) AND
                    ¬ TERMINOU DO
          BEGIN
           SIMBOL:= SIMBPROD(POSSIMBOL);
           IF SIMBOL ETERMINAL
            THEN
             BEGIN
              ADDFIRST(NAOTERMINAL,SIMBOL);
              TERMINOU:= TRUE;
             END
            ELSE
             BEGIN
              ADDLINHAFIRST(SIMBOL,NACTERMINAL);
              IF ¬GERAEPSILON(SIMBOL)
               THEN TERMINOU:= TRUE;
             END;
          END;
         IF ¬TERMINOU
          THEN
           BEGIN
            ADDFIRSTEPSILON(NACTERMINAL);
           END;
       END;
     END;
   END;
  END;
END GERAFIRST;
%*************************************************************************
% ROTINA DE IMPRESSAO DA RELACAO FIRST  PARA OS SIMBOLOS NAO TERMINAIS*
% DA GRAMATICA DE ENTRADA                                               *
%*************************************************************************
%                              ****************************
PROCEDURE IMPFIRST;
BEGIN
 INTEGER
   CONTTERM,
   CONTLINHA,
   NAOTERMINAL,
   TERMINAL,
   I;
```

```
 BOOLEAN
   INICIONOVONAOTERMINAL;
 INTEGER ARRAY
   MATTERMLINHA 0:16 ;
PROCEDURE CABECALHO;
BEGIN
 WRITE(IMPRE1 SKIP 1 );
 WRITE(IMPRE1,<X64,"FIRST">);
 WRITE(IMPRE1 SPACE 1 ,<X56,"TABELA RESULTADO">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 WRITE(IMPRE1 SPACE 1 ,<X3,"NAO  |">);
 WRITE(IMPRE1 SPACE 1 ,<   "TERMINAL|",X47,"LISTA DE TERMINAIS">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 CONTLINHA:= 0;
END CABECALHO;
 CABECALHO;
 FOR NAOTERMINAL:= PRIMEIRONAOTERMINAL STEP 1 UNTIL
                   ULTIMONAOTERMINAL DO
  BEGIN
   INICIONOVONAOTERMINAL:= TRUE;
   CONTTERM:= 0;
   FOR TERMINAL:= PRIMEIROTERMINAL STEP 1 UNTIL ULTIMOTERMINAL DO
     IF FIRST(NAOTERMINAL,TERMINAL)
      THEN
       BEGIN
        CONTTERM:= CONTTERM + 1;
        MATTERMLINHA RESTO(CONTTERM - 1, 17) := TERMINAL;
        IF RESTO(CONTTERM,17) = 0
         THEN
          IF INICIONOVONAOTERMINAL
           THEN
            BEGIN
             IF CONTLINHA > 75
              THEN CABECALHO;
             WRITE(IMPRE1 SPACE 1 );
             WRITE(IMPRE1          ,<I6,X4,17(I6,X1)>,NAOTERMINAL,
                    FOR I:= 0 STEP 1 UNTIL 16 DO MATTERMLINHA I );
             INICIONOVONAOTERMINAL:= FALSE;
             CONTLINHA:= CONTLINHA + 2;
            END
```

```
            ELSE
             BEGIN
              IF CONTLINHA > 75
               THEN CABECALHO;
              WRITE(IMPRE1          ,<X10,17(I6,X1)>,
                FOR I:= 0 STEP 1 UNTIL 16 DO MATTERMLINHA I );
              CONTLINHA:= CONTLINHA + 1;
             END;
        END;
    IF RESTO(CONTTERM,17)¬= 0
     THEN
      IF INICIONOVONAOTERMINAL
       THEN
        BEGIN
         IF CONTLINHA > 75
          THEN CABECALHO;
         WRITE(IMPRE1 SPACE 1 );
         WRITE(IMPRE1          ,<I6,X4,17(I6,X1)>,NAOTERMINAL,
             FOR I:= 0 STEP 1 UNTIL RESTO(CONTTERM-1,17) DO
                       MATTERMLINHA I );
         CONTLINHA:= CONTLINHA + 2;
        END
       ELSE
        BEGIN
         IF CONTLINHA > 75
          THEN CABECALHO;
         WRITE(IMPRE1          ,<X10,17(I6,X1)>,FOR I:= 0 STEP 1
                  UNTIL RESTO(CONTTERM-1,17) DO MATTERMLINHA I );
         CONTLINHA:= CONTLINHA + 1;
        END;
    IF GERAEPSILON(NAOTERMINAL)
     THEN
      BEGIN
       IF CONTLINHA > 75
        THEN CABECALHO;
       WRITE(IMPRE1          ,<X10,"EPSILON">);
       CONTLINHA:= CONTLINHA + 1;
      END;
   END;
END IMPFIRST;
```

```
%                              ****************************
%*************************************************************************
% ROTINA DE CALCULO DA RELACAO FOLLOW PARA OS SIMBOLOS NAC TERMINAIS   *
% DA GRAMATICA DE ENTRADA                                              *
%*************************************************************************
PROCEDURE GERAFOLLOW;
BEGIN
 INTEGER
   NACTERMINAL,
   PRODUCAC,
   POSSIMBOL,
   SIMBOL,
   POSSIMBDIR,
   SIMBDIR;
 BOOLEAN
   ADD,
   TERMINOU;
PROCEDURE ADDLINHAFOLLOW(NAOTERM1,NACTERM2);
INTEGER NACTERM1,NACTERM2;
BEGIN
 INTEGER
   INDAUX;
 REAL
   VARAUX;
 FOR INDAUX:= 0 STEP 1 UNTIL INDMAX DO
  BEGIN
   VARAUX:= REAL(MATFOLLOW NACTERM2,INDAUX );
   MATFOLLOW NACTERM2,INDAUX := MATFOLLOW NACTERM2,INDAUX  OR
                                MATFOLLOW NACTERM1,INDAUX ;
   IF VARAUX. 47:48  ¬= REAL(MATFOLLOW NACTERM2,INDAUX )
    THEN ADD:= TRUE;
  END;
END ADDLINHAFOLLOW;
PROCEDURE ADDLINHAFIRSTFOLLOW(NACTERM1,NACTERM2);
INTEGER NACTERM1,NACTERM2;
BEGIN
 INTEGER
   INDAUX;
 REAL
   VARAUX;
```

```
 FOR INDAUX:= 0 STEP 1 UNTIL INDMAX DO
  BEGIN
   VARAUX:= REAL(MATFOLLOW NAOTERM2,INDAUX );
   MATFOLLOW NAOTERM2,INDAUX := MATFOLLOW NAOTERM2,INDAUX  OR
                                MATFIRST NAOTERM1,INDAUX ;
   IF VARAUX. 47:48  ¬= REAL(MATFOLLOW NAOTERM2,INDAUX )
    THEN ADD:= TRUE;
  END;
END ADDLINHAFIRSTFOLLOW;
PROCEDURE ADDFOLLOW(SIMBOL,TERM);
INTEGER SIMBOL,TERM;
BEGIN
 IF ¬FOLLOW(SIMBOL,TERM)
  THEN
   BEGIN
    FOLLOW(SIMBOL,TERM):= TRUE;
    ADD:= TRUE;
   END;
END ADDFOLLOW;
 ADDFOLLOW(0,DOLLAR);
 ADD:= TRUE;
 WHILE ADD DO
  BEGIN
   ADD:= FALSE;
   FOR NAOTERMINAL:= 0 STEP 1 UNTIL
                     ULTIMONAOTERMINAL DO
    FOR PRODUCAO:= PRIMEIRAPRODUCAO(NAOTERMINAL) STEP 1 UNTIL
                     ULTIMAPRODUCAO(NAOTERMINAL) DO
     BEGIN
      FOR POSSIMBOL:= POSPRIMSIMBOL(PRODUCAO) STEP 1 UNTIL
                        POSULTSIMBOL(PRODUCAO) DO
       BEGIN
        SIMBOL:= SIMBPROD(POSSIMBOL);
        IF SIMBOL ENAOTERMINAL
         THEN
          IF POSSIMBOL = POSULTSIMBOL(PRODUCAO)
           THEN
            ADDLINHAFOLLOW(NAOTERMINAL,SIMBOL)
           ELSE
            BEGIN
```

```
                TERMINOU:= FALSE;
                FOR POSSIMBDIR:= POSSIMBOL+1 STEP 1 WHILE
                                 POSSIMBDIR <= POSULTSIMBOL(PRODUCAO)
                                 AND ¬TERMINOU DO
                 BEGIN
                  SIMBDIR:= SIMBPROD(POSSIMBDIR);
                  IF SIMBDIR ETERMINAL
                   THEN
                    BEGIN
                     ADDFOLLOW(SIMBOL,SIMBDIR);
                     TERMINOU:= TRUE;
                    END
                   ELSE
                    BEGIN
                     ADDLINHAFIRSTFOLLOW(SIMBDIR,SIMBOL);
                     IF ¬GERAEPSILON(SIMBDIR)
                      THEN TERMINOU:= TRUE;
                     END;
                 END;
                IF ¬TERMINOU
                 THEN
                  BEGIN
                   ADDLINHAFOLLOW(NAOTERMINAL,SIMBOL);
                  END;
               END;
        END;
      END;
  END;
END GERAFOLLOW;
%                              ******************************
%*****************************************************************************
% ROTINA DE IMPRESSAO DA RELACAO FOLLOW PARA OS SIMBOLOS NAO TERMINAIS*
% DA GRAMATICA DE ENTRADA                                              *
%*****************************************************************************
PROCEDURE IMPFOLLOW;
BEGIN
 INTEGER
   CONTTERM,
   CONTLINHA,
   NAOTERMINAL,
```

```
    TERMINAL,
    I;
  BOOLEAN
    INICIONOVONAOTERMINAL;
  INTEGER ARRAY
    MATTERMLINHA 0:16 ;
PROCEDURE CABECALHO;
BEGIN
 WRITE(IMPRE1 SKIP 1 );
 WRITE(IMPRE1,<X64,"FOLLOW">);
 WRITE(IMPRE1 SPACE 1 ,<X56,"TABELA RESULTADO">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 WRITE(IMPRE1 SPACE 1 ,<X3,"NAO  |">);
 WRITE(IMPRE1 SPACE 1 ,<   "TERMINAL|",X47,"LISTA DE TERMINAIS">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 CONTLINHA:= 0;
END CABECALHO;
  CABECALHO;
 FOR NAOTERMINAL:= 0 STEP 1 UNTIL
                   ULTIMONAOTERMINAL DO
  BEGIN
   INICIONOVONAOTERMINAL:= TRUE;
   CONTTERM:= 0;
   FOR TERMINAL:= PRIMEIROTERMINAL STEP 1 UNTIL ULTIMOTERMINAL DO
     IF FOLLOW(NAOTERMINAL,TERMINAL)
      THEN
       BEGIN
        CONTTERM:= CONTTERM + 1;
        MATTERMLINHA RESTO(CONTTERM - 1, 17) := TERMINAL;
        IF RESTO(CONTTERM,17) = 0
         THEN
          IF INICIONOVONAOTERMINAL
           THEN
            BEGIN
             IF CONTLINHA > 75
              THEN CABECALHO;
             WRITE(IMPRE1 SPACE 1 );
             WRITE(IMPRE1          ,<I6,X4,17(I6,X1)>,NAOTERMINAL,
                    FOR I:= 0 STEP 1 UNTIL 16 DO MATTERMLINHA I );
             INICIONOVONAOTERMINAL:= FALSE;
```

```
            CONTLINHA:= CONTLINHA + 2;
           END
          ELSE
           BEGIN
            IF CONTLINHA > 75
             THEN CABECALHO;
            WRITE(IMPRE1          ,<X10,17(I6,X1)>,
              FOR I:= 0 STEP 1 UNTIL 16 DO MATTERMLINHA I );
            CONTLINHA:= CONTLINHA + 1;
           END;
       END;
   IF RESTO(CONTTERM,17)¬= 0
    THEN
     IF INICIONOVONAOTERMINAL
      THEN
       BEGIN
        IF CONTLINHA > 75
         THEN CABECALHO;
        WRITE(IMPRE1 SPACE 1 );
        WRITE(IMPRE1          ,<I6,X4,17(I6,X1)>,NAOTERMINAL,
            FOR I:= 0 STEP 1 UNTIL RESTO(CONTTERM-1,17) DO
                     MATTERMLINHA I );
        CONTLINHA:= CONTLINHA + 2;
       END
      ELSE
       BEGIN
        IF CONTLINHA > 75
         THEN CABECALHO;
        WRITE(IMPRE1          ,<X10,17(I6,X1)>,FOR I:= 0 STEP 1
                UNTIL RESTO(CONTTERM-1,17) DO MATTERMLINHA I );
        CONTLINHA:= CONTLINHA + 1;
       END;
  END;
END IMPFOLLOW;
%                          ****************************
%*************************************************************************
% ROTINA DE IMPRESSAO DAS PRODUCOES COM OS SIMBOLOS IMPRESSOS NAS SUAS*
% REPRESENTACOES INTERNAS                                               *
%*************************************************************************
PROCEDURE IMPPRODUCOES;
```

```
BEGIN
 INTEGER
   CONTSIMB,
   CONTLINHA,
   NAOTERMINAL,
   PRODUCAO,
   POSSIMBOL,
   I;
 BOOLEAN
   INICIONOVAPRODUCAO;
 INTEGER ARRAY
   MATSIMBLINHA 0:15 ;
PROCEDURE CABECALHO;
BEGIN
 WRITE(IMPRE1 SKIP 1 );
 WRITE(IMPRE1,<X64,"PRODUCOES">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 WRITE(IMPRE1 SPACE 1 ,<"         |   NAO   |">);
 WRITE(IMPRE1 SPACE 1 ,<"PRODUCAO|TERMINAL|",X42,"LISTA DE SIMBOLOS">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 CONTLINHA:= 0;
END CABECALHO;
  CABECALHO;
 FOR NAOTERMINAL:= 0 STEP 1 UNTIL
                   ULTIMONAOTERMINAL DO
  FOR PRODUCAO:= PRIMEIRAPRODUCAO(NAOTERMINAL) STEP 1 UNTIL
                   ULTIMAPRODUCAO(NAOTERMINAL) DO
  IF PRODUCAOEPSILON(PRODUCAO)
   THEN
    BEGIN
     IF CONTLINHA > 75
      THEN CABECALHO;
     WRITE(IMPRE1 SPACE 1 );
     WRITE(IMPRE1 SPACE 1 ,<I6,X3,I6," ->",X2,"EPSILON">,PRODUCAO
           ,NAOTERMINAL);
     CONTLINHA:= CONTLINHA + 2;
    END
   ELSE
  BEGIN
   INICIONOVAPRODUCAO:= TRUE;
```

```
CONTSIMB:= 0;
FOR POSSIMBOL:= POSPRIMSIMBOL(PRODUCAO) STEP 1 UNTIL
                POSULTSIMBOL(PRODUCAO) DO
    BEGIN
     CONTSIMB:= CONTSIMB + 1;
     MATSIMBLINHA RESTO(CONTSIMB - 1,16) := SIMBPROD(POSSIMBOL);
     IF RESTO(CONTSIMB,16) = 0
      THEN
       IF INICIONOVAPRODUCAO
        THEN
         BEGIN
          IF CONTLINHA > 75
           THEN CABECALHO;
          WRITE(IMPRE1 SPACE 1 );
          WRITE(IMPRE1          ,<I6,X3,I6," ->",16(X1,I6)>,PRODUCAO,
                                          NAOTERMINAL,
                 FOR I:= 0 STEP 1 UNTIL 15 DO MATSIMBLINHA I );
          INICIONOVAPRODUCAO:= FALSE;
          CONTLINHA:= CONTLINHA + 2;
         END
        ELSE
         BEGIN
          IF CONTLINHA > 75
           THEN CABECALHO;
          WRITE(IMPRE1          ,<X19,16(X1,I6)>,
            FOR I:= 0 STEP 1 UNTIL 15 DO MATSIMBLINHA I );
          CONTLINHA:= CONTLINHA + 1;
         END;
    END;
IF RESTO(CONTSIMB,16)¬= 0
 THEN
  IF INICIONOVAPRODUCAO
   THEN
    BEGIN
     IF CONTLINHA > 75
      THEN CABECALHO;
     WRITE(IMPRE1 SPACE 1 );
     WRITE(IMPRE1          ,<I6,X3,I6," ->",16(X1,I6)>,PRODUCAO,
                              NAOTERMINAL,
         FOR I:= 0 STEP 1 UNTIL RESTO(CONTSIMB-1,16) DO
```

```
                        MATSIMBLINHA I );
          CONTLINHA:= CONTLINHA + 2;
         END
        ELSE
         BEGIN
          IF CONTLINHA > 75
           THEN CABECALHC;
          WRITE(IMPRE1          ,<X19,16(X1,I6)>,FOR I:= 0 STEP 1
                    UNTIL RESTO(CONTSIMB-1,16) DO MATSIMBLINHA I );
          CONTLINHA:= CONTLINHA + 1;
         END;
   END;
END IMPPRODUCCES;
%                              ***************************
%*****************************************************************************
% ROTINA QUE ELIMINA UM ITEM DO FECHAMENTO DE UM CONJUNTO DE ITENS     *
% LR(0) OU LR(1)                                                      *
%*****************************************************************************
PROCEDURE ELIMINACLOS(ITEM);
 INTEGER ITEM;
 BEGIN
  IF ITEM = PRIMITEMCLOS
   THEN
    IF PROXITEMCLOS(ITEM) ¬= TERRA
     THEN
       PRIMITEMCLOS:= PROXITEMCLOS(ITEM)
     ELSE
       PRIMITEMCLOS:= TERRA
   ELSE
    BEGIN
     PROXITEMCLOS(ITEMCLOSANT(ITEM)):= PROXITEMCLOS(ITEM);
     IF PROXITEMCLOS(ITEM) ¬= TERRA
      THEN
       ITEMCLOSANT(PROXITEMCLOS(ITEM)):= ITEMCLOSANT(ITEM);
    END;
END ELIMINACLOS;
%                              ***************************
%*****************************************************************************
 % PROCESSAMENTO SLR(1) E LALR(1)                                            *
%*****************************************************************************
```

```
PROCEDURE SLRLALR;
BEGIN
 INTEGER ARRAY
   MATCONJITEM 0:SIZECITEM ,
   MATITEM 0:SIZEITEM ,
   MATGOTO 0:SIZECITEM,1:NUMNAOTERM+NUMTERM ;
 INTEGER
   CITEMMAX;
 DEFINE
   POSPRIMITEMLRO(CITEM)=MATCONJITEM CITEM . 47:16 #,
    POSULTITEMLRO(CITEM)=MATCONJITEM CITEM . 47:16  +
                         MATCONJITEM CITEM . 31:16  - 1#,
   PRODITEMLRO(PITEM)=MATITEM PITEM . 47:16 #,
   POSPONTOITEMLRO(PITEM)=MATITEM PITEM . 31:16 #,
   TAMCONJITEM(CITEM)=MATCONJITEM CITEM . 31:16 #,
   PRIMCONJITEM=0#,
   ULTCONJITEM=CITEMMAX#,
   GOTO(CITEM,SIMBOL)=MATGOTO CITEM,SIMBOL #;
%*****************************************************************
% ROTINA DE CALCULO DO FECHAMENTO DE UM CONJUNTO DE ITENS LR(0)    *
%*****************************************************************
PROCEDURE CLOSURELRO(CITEM);
INTEGER CITEM;
BEGIN
 INTEGER
   POSITEM,
   PANA,
   SIMBOL,
   PRODUCAC;
 BOOLEAN
  ARRAY
          ADD 0:ENTIER((NUMNAOTERM-1)/48) ;
 DEFINE
   ADDCLOSLRO(SIMB)=ADD ENTIER((SIMB-1)/48) .
                        (SIMB-1 - 48*ENTIER((SIMB-1)/48)):1 #;
 PRIMITEMCLOS:= 1;
 NUMITEMCLOS:= 1;
 PRODITEMCLOS(NUMITEMCLOS):= PRODITEMLRO(POSPRIMITEMLRO(CITEM));
 POSPONTOITEMCLOS(NUMITEMCLOS):= POSPONTOITEMLRO(POSPRIMITEMLRO(
                                                      CITEM));
```

```
 ITEMCLOSANT(NUMITEMCLOS):= TERRA;
 FOR POSITEM:= POSPRIMITEMLRO(CITEM) + 1 STEP 1 UNTIL
                POSULTITEMLRO(CITEM) DO
  BEGIN
   NUMITEMCLOS:= NUMITEMCLOS+1;
   PRODITEMCLOS(NUMITEMCLOS):= PRODITEMLRO(POSITEM);
   POSPONTOITEMCLOS(NUMITEMCLOS):= POSPONTOITEMLRO(POSITEM);
   PROXITEMCLOS(NUMITEMCLOS-1):= NUMITEMCLOS;
   ITEMCLOSANT(NUMITEMCLOS):= NUMITEMCLOS - 1;
  END;
 PANA:= 1;
 WHILE PANA <= NUMITEMCLOS DO
  BEGIN
   IF POSPONTOITEMCLOS(PANA) < TAMPRODUCAO(PRODITEMCLOS(PANA))
    THEN
     BEGIN
     SIMBOL:= SIMBPROD(POSSIMBPONTOITEMCLOS(PANA));
     IF SIMBOL ENAOTERMINAL
     THEN
     IF ¬ADDCLOSLRO(SIMBOL)
      THEN
       BEGIN
        FOR PRODUCAO:= PRIMEIRAPRODUCAO(SIMBOL) STEP 1 UNTIL
                       ULTIMAPRODUCAO(SIMBOL) DO
         BEGIN
          NUMITEMCLOS:= NUMITEMCLOS+1;
          PRODITEMCLOS(NUMITEMCLOS):= PRODUCAO;
          POSPONTOITEMCLOS(NUMITEMCLOS):= 0;
          PROXITEMCLOS(NUMITEMCLOS-1):= NUMITEMCLOS;
          ITEMCLOSANT(NUMITEMCLOS):= NUMITEMCLOS - 1;
         END;
        ADDCLOSLRO(SIMBOL):= TRUE;
       END;
     END;
   PANA:= PANA + 1;
  END;
    PROXITEMCLOS(NUMITEMCLOS):= TERRA;
END CLOSURELRO;
%                              ***************************
%*************************************************************************
```

```
% ROTINA DE GERACAO DOS CONJUNTOS DE ITENS LR(0)                    *
%*******************************************************************
PROCEDURE GERCONJITEMLRO;
BEGIN
 INTEGER
   CITEMAN;
%*******************************************************************
% ROTINA DE CALCULO DOS CONJUNTOS DE ITENS POSSIVEIS NO GOTO DO     *
% CONJUNTO DE ITEM CITEM                                            *
%*******************************************************************
PROCEDURE GERAGOTO(CITEM);
INTEGER CITEM;
BEGIN
 INTEGER
   SIMBOL,
   NITEM,
   ITEM,
   ITEMANT,
   CITEMENCONTRADO;
 BOOLEAN
   ENCONTROU;
 DEFINE
   POSINSPRIMITEM=POSULTITEMLRO(CITEMMAX-1) + 1#,
   POSINSITEM=POSULTITEMLRO(CITEMMAX) + NITEM + 1#;
PROCEDURE PROCURAITEMIDENTICO(ENCONTROU,CITEMID);
BOOLEAN ENCONTROU;
INTEGER CITEMID;
BEGIN
 INTEGER
   POS1,
   IND1,
   IND2,
   INTAUX,
   CONJITEM;
 POS1:= POSULTITEMLRO(CITEMMAX);
 FOR IND1:= POS1+NITEM-1 STEP -1 UNTIL POS1+1 DO
  FOR IND2:= POS1+1 STEP 1 UNTIL IND1 DO
   IF MATITEM IND2+1  . 47:48 < MATITEM IND2 . 47:48
    THEN
     BEGIN
```

```
      INTAUX. 47:48 := MATITEM IND2 . 47:48 ;
      MATITEM IND2 . 47:48 := MATITEM IND2+1 . 47:48 ;
      MATITEM IND2+1 . 47:48 := INTAUX. 47:48 ;
      END;
  ENCONTROU:= FALSE;
  FOR CONJITEM:= 1 STEP 1 WHILE CONJITEM <= CITEMMAX AND
                                    ¬ ENCONTROU DO
   BEGIN
    IF TAMCONJITEM(CONJITEM) = NITEM
      THEN
       BEGIN
        FOR IND1:= 0 STEP 1 WHILE MATITEM POSPRIMITEMLRO(CONJITEM)+IND1
      . 47:48 = MATITEM POS1+1+IND1 . 47:48  AND IND1 < NITEM DO;
        IF IND1 = NITEM
         THEN ENCONTROU:= TRUE;
       END;
   END;
  CITEMID:= CONJITEM - 1;
END PROCURACITEMIDENTICO;
  CLOSURELRO(CITEM);
  WHILE PRIMITEMCLOS ¬= TERRA DO
   BEGIN
    NITEM:= 0;
    IF TAMPRODUCAO(PRODITEMCLOS(PRIMITEMCLOS)) >
       POSPONTOITEMCLOS(PRIMITEMCLOS)
     THEN
      BEGIN
       SIMBOL:= SIMBPROD(POSSIMBPONTOITEMCLOS(PRIMITEMCLOS));
       PRODITEMLRO(POSINSITEM):= PRODITEMCLOS(PRIMITEMCLOS);
       POSPONTOITEMLRO(POSINSITEM):= POSPONTOITEMCLOS(PRIMITEMCLOS)+1;
       NITEM:= NITEM + 1;
       ELIMINACLOS(PRIMITEMCLOS);
       ITEM:= PRIMITEMCLOS;
       WHILE ITEM ¬= TERRA DO
        BEGIN
         IF TAMPRODUCAO(PRODITEMCLOS(ITEM)) >
            POSPONTOITEMCLOS(ITEM)
          THEN
           BEGIN
           IF SIMBPROD(POSSIMBPONTOITEMCLOS(ITEM)) = SIMBOL
```

```
            THEN
             BEGIN
              PRODITEMLRO(POSINSITEM):= PRODITEMCLOS(ITEM);
              POSPONTOITEMLRO(POSINSITEM):= POSPONTOITEMCLOS(ITEM) + 1;
              NITEM:= NITEM + 1;
              ELIMINACLOS(ITEM);
             END;
           END
          ELSE
           ELIMINACLOS(ITEM);
         ITEM:= PROXITEMCLOS(ITEM);
        END;
    IF NITEM > 0
     THEN
      BEGIN
       PROCURACITEMIDENTICO(ENCONTROU,CITEMENCONTRADO);
       IF - ENCONTROU
        THEN
         BEGIN
          CITEMMAX:= CITEMMAX + 1;
          POSPRIMITEMLRO(CITEMMAX):= POSINSPRIMITEM;
          TAMCONJITEM(CITEMMAX):= NITEM;
          GOTO(CITEM,SIMBOL):= CITEMMAX;
         END
        ELSE
          GOTO(CITEM,SIMBOL):= CITEMENCONTRADO;
      END;
     END
    ELSE
     ELIMINACLOS(PRIMITEMCLOS);
  END;
END GERAGOTO;
%                          **************************
 POSPRIMITEMLRO(0):= 1;
 TAMCONJITEM(0):= 1;
 PRODITEMLRO(1):= 0;
 POSPONTOITEMLRO(1):= 0;
 WHILE CITEMAN <= CITEMMAX DO
  BEGIN
   GERAGOTO(CITEMAN);
```

```
    CITEMAN:= CITEMAN + 1;
   END;
END GERCONJITEMLRO;
%                              ************************
%*************************************************************************
% ROTINA DE IMPRESSAO DA RELACAO GOTO PARA CONJUNTOS DE ITENS LR(0)    *
% OU LALR(1)                                                           *
%*************************************************************************
PROCEDURE IMPGOTOLRO;
BEGIN
 INTEGER
   CONJITEM,
   CONTPAR,
   NACTERMINAL,
   CONTLINHA,
   I;
 BOOLEAN
   INICIONOVOCITEM;
 INTEGER
  ARRAY
   MATNTERMLINHA 0:6 ,
   MATCITEMLINHA 0:6 ;
PROCEDURE CABECALHO;
BEGIN
 WRITE(IMPRE1 SKIP 1 );
 WRITE(IMPRE1,<X59,"GOTO (LRO)">);
 WRITE(IMPRE1 SPACE 1 ,<X56, "TABELA RESULTADO">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 WRITE(IMPRE1 SPACE 1 ,<"CONJUNTO|", 7("  NAC  CONJUNTO ")>);
 WRITE(IMPRE1 SPACE 1 ,<"    DE  |", 7("  TER    ITEM  |")>);
 WRITE(IMPRE1 SPACE 1 ,<"   ITEM |", 7(" MINAL   FINAL |")>);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 CONTLINHA:= 0;
END CABECALHO;
 CABECALHO;
 FOR CONJITEM:= PRIMCONJITEM STEP 1 UNTIL ULTCONJITEM DO
  BEGIN
   INICIONOVOCITEM:= TRUE;
   CONTPAR:= 0;
   FOR NACTERMINAL:= PRIMEIRONACTERMINAL STEP 1
```

```
                                        UNTIL ULTIMONAOTERMINAL DO
 IF GOTO(CONJITEM,NAOTERMINAL) ¬= 0
  THEN
   BEGIN
    CONTPAR:= CONTPAR + 1;
    MATNTERMLINHA RESTO(CONTPAR-1,7) := NAOTERMINAL;
    MATCITEMLINHA RESTO(CONTPAR-1,7) := GOTO(CONJITEM,NAOTERMINAL);
    IF RESTO(CONTPAR,7) = 0
     THEN
      IF INICIONOVOCITEM
       THEN
        BEGIN
         IF CONTLINHA > 75
          THEN CABECALHO;
         WRITE(IMPRE1 SPACE 1 );
         WRITE(IMPRE1,<I8,X1,7(I6,X1,I8,"|")>,CONJITEM,
            FOR I:= 0 STEP 1 UNTIL 6 DO MATNTERMLINHA I ,
                                        MATCITEMLINHA I  );
         INICIONOVOCITEM:= FALSE;
         CONTLINHA:= CONTLINHA + 2;
        END
       ELSE
        BEGIN
         IF CONTLINHA > 75
          THEN CABECALHO;
         WRITE(IMPRE1,<X9,7(I6,X1,I8,"|")>,
            FOR I:= 0 STEP 1 UNTIL 6 DO MATNTERMLINHA I ,
                                        MATCITEMLINHA I  );
         CONTLINHA:= CONTLINHA + 1;
        END;
   END;
IF RESTO(CONTPAR,7) ¬= 0
 THEN
  IF INICIONOVOCITEM
   THEN
    BEGIN
     IF CONTLINHA > 75
      THEN CABECALHO;
     WRITE(IMPRE1 SPACE 1 );
     WRITE(IMPRE1,<I8,X1,7(I6,X1,I8,"|")>,CONJITEM,
```

```
          FOR I:= 0 STEP 1 UNTIL RESTO(CONTPAR-1,7) DO
                                        MATNTERMLINHA I ,
                                        MATCITEMLINHA I  );
         CONTLINHA:= CONTLINHA + 2;
        END
       ELSE
        BEGIN
         IF CONTLINHA > 75
          THEN CABECALHO;
         WRITE(IMPRE1,<X9,7(I6,X1,I8,"|")> ,
           FOR I:= 0 STEP 1 UNTIL RESTO(CONTPAR-1,7) DO
                                        MATNTERMLINHA I ,
                                        MATCITEMLINHA I  );
         CONTLINHA:= CONTLINHA + 1;
        END;
  END;
END IMPGOTOLRO;
%                              ***************************
%*************************************************************************
% ROTINA DE GERACAO DA TABELA DE ANALISE SINTATICA SLR(1)                 *
%*************************************************************************
PROCEDURE GERACTIONSLR;
BEGIN
 INTEGER
   CONJITEM,
   TERMINAL,
   ITEM,
   CONTLINHACONFLITO,
   CONTTER,
   CONTLINHAGLRO,
   I,
   CONTTERPARSER,
   CONTLINHAPSLR,
   SIMBOL;
 BOOLEAN
   INICIONOVOCITEM,
   INICIONOVOCITEMPARSER;
 INTEGER
  ARRAY
   MATPROLINHA 0:10 ,
```

```
    MATPPCNLINHA 0:10 ,
    MATTERMLINHA 0:4 ,
    MATACACLINHA 0:4 ,
    MATCAUXLINHA 0:4 ,
    ACTIONSLR NUMNAOTERM+1:NUMNAOTERM+NUMTERM ;
  DEFINE
    INICIALIZARLINHATABELASLR=
    FOR TERMINAL:= PRIMEIROTERMINAL STEP 1 UNTIL ULTIMOTERMINAL DO
     ACTIONSLR TERMINAL := 0#;
PROCEDURE CABECALHOPSLR;
BEGIN
 WRITE(IMPRE2 SKIP 1 );
 WRITE(IMPRE2,<X56,"GERADOR PARSER SLR">);
 WRITE(IMPRE2 SPACE 1 ,<X57,"TABELA RESULTADO">);
 WRITE(IMPRE2 SPACE 1 ,<132("-")>);
 WRITE(IMPRE2 SPACE 1 ,<"        |",5("  TER          ESTADO |")>);
 WRITE(IMPRE2 SPACE 1 ,<"  ESTADO|",5(" MINAL  ACAO  PRODUCAO|")>);
 WRITE(IMPRE2 SPACE 1 ,<132("-")>);
 CONTLINHAPSLR:= 0
END CABECALHOPSLR;
PROCEDURE CABECALHOCLRO;
BEGIN
 WRITE(IMPRE1 SKIP 1 );
 WRITE(IMPRE1,<X52,"CONJUNTOS DE ITENS LRO">);
 WRITE(IMPRE1 SPACE 1 ,<X56,"TABELA RESULTADO">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 WRITE(IMPRE1 SPACE 1 ,<"CONJUNTO|",11("       POS|")>);
 WRITE(IMPRE1 SPACE 1 ,<"   DE   |",11(" PRODU PON|")>);
 WRITE(IMPRE1 SPACE 1 ,<"  ITENS |",11("  CAO   TO|")>);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 CONTLINHACLRO:= 0;
END CABECALHOCLRO;
PROCEDURE IMPCONFLITOSLR(TERM,TIPO,NUM);
INTEGER TERM,TIPO,NUM;
BEGIN
PROCEDURE CABECALHO;
BEGIN
 WRITE(IMPRE3 SKIP 1 );
 WRITE(IMPRE3,<X26,"GERADOR PARSER SLR">);
 WRITE(IMPRE3 SPACE 1 ,<X31,"CONFLITOS">);
```

```
 WRITE(IMPRE3 SPACE 1 ,<70("-")>);
 WRITE(IMPRE3 SPACE 1 ,<X24,18("-"),X4,18("-")>);
 WRITE(IMPRE3 SPACE 1 ,<X4,"ESTADO",X3,"TERMINAL",2(X14,"ESTADO",X2)>);
 WRITE(IMPRE3 SPACE 1 ,<X24,2(" ACAO     PRODUCAO    ")>);
 WRITE(IMPRE3 SPACE 1 ,<70("-")>);
 CONTLINHACONFLITO:= 0;
END CABECALHO;
 IF CONTLINHACONFLITO > 75
  THEN CABECALHO;
 WRITE(IMPRE3 SPACE 1 );
 WRITE(IMPRE3,<X2,I8,X4,I6,X4,A6,X4,I8,X4,A6,X4,I8>,
         CONJITEM,TERM,NACT TIPO ,NUM,NACT ACTIONSLR TERM . 47:2  ,
          ACTIONSLR TERM . 45:46 );
 CONTLINHACONFLITO:= CONTLINHACONFLITO + 1;
END IMPCONFLITOSLR;
PROCEDURE ADDACTIONSLR(TERM,TIPO,NUM);
INTEGER TERM,TIPO,NUM ;
BEGIN
 IF ACTIONSLR TERM . 47:2  ¬= 0
  THEN
   BEGIN
    IF ¬ (ACTIONSLR TERM . 47:2  = TIPO AND
          ACTIONSLR TERM . 45:46 =  NUM)
     THEN
      IMPCONFLITOSLR(TERM,TIPO,NUM);
   END
  ELSE
   BEGIN
    ACTIONSLR TERM . 47:2  := TIPO;
    ACTIONSLR TERM . 45:46 := NUM;
    CONTTERPARSER:= CONTTERPARSER + 1;
    MATTERMLINHA RESTO(CONTTERPARSER-1,5) := TERM;
    MATACAOLINHA RESTO(CONTTERPARSER-1,5) := TIPO;
    MATCAUXLINHA RESTO(CONTTERPARSER-1,5) := NUM;
    IF RESTO(CONTTERPARSER,5) = 0
     THEN
      IF INICIONOVOCITEMPARSER
       THEN
        BEGIN
         IF CONTLINHAPSLR > 75
```

```
          THEN CABECALHOPSLR;
         WRITE(IMPRE2 SPACE 1 );
         WRITE(IMPRE2,<I8,X1,5(I6,X1,A6,X1,I8,"|")>,
           CONJITEM,FOR I:= 0 STEP 1 UNTIL 4 DO MATTERMLINHA I ,
                                              NACT MATACAOLINHA I  ,
                                                   MATCAUXLINHA I  );
         INICIONOVOCITEMPARSER:= FALSE;
         CONTLINHAPSLR:= CONTLINHAPSLR + 2;
        END
       ELSE
         BEGIN
          IF CONTLINHAPSLR >75
           THEN CABECALHOPSLR;
          WRITE(IMPRE2,<X9,5(I6,X1,A6,X1,I8,"|")>,
            FOR I:= 0 STEP 1 UNTIL 4 DO MATTERMLINHA I ,
                                     NACT MATACAOLINHA I  ,
                                          MATCAUXLINHA I  );
          CONTLINHAPSLR:= CONTLINHAPSLR + 1;
         END;
  END;
END ACDACTIONSLR;
 CABECALHOCLRO;
 CABECALHOPSLR;
 CONTLINHACONFLITO:= 76;
 FOR CONJITEM:= PRIMCONJITEM STEP 1 UNTIL ULTCONJITEM DO
  BEGIN
   INICIONOVOCITEM:= TRUE;
   CONTTER:= 0;
   INICIONOVOCITEMPARSER:= TRUE;
   CONTTERPARSER:= 0;
   INICIALIZARLINHATABELASLR;
   CLOSURELRO(CONJITEM);
   FOR ITEM:= 1 STEP 1 UNTIL NUMITEMCLOS DO
       BEGIN
        CONTTER:= CONTTER + 1 ;
        MATPRODLINHA RESTO(CONTTER-1,11) := PRODITEMCLOS(ITEM);
        MATPPONLINHA RESTO(CONTTER-1,11) := POSPONTOITEMCLOS(ITEM);
        IF RESTO(CONTTER,11) = 0
         THEN
          IF INICIONOVOCITEM
```

```
        THEN
         BEGIN
          IF CONTLINHACLRO > 75
           THEN CABECALHOCLRO;
          WRITE(IMPRE1 SPACE 1 );
          WRITE(IMPRE1,<I8,X1,11(I6,X1,I3,"|")>,
            CONJITEM,FOR I:= 0 STEP 1 UNTIL 10 DO MATPRODLINHA I ,
                                                  MATPPONLINHA I  );
          INICIONOVOCITEM:= FALSE;
          CONTLINHACLRO := CONTLINHACLRO + 2;
         END
        ELSE
         BEGIN
          IF CONTLINHACLRO > 75
           THEN CABECALHOCLRO;
          WRITE(IMPRE1,<X9, 11(I6,X1,I3,"|")>,
            FOR I:= 0 STEP 1 UNTIL 10 DO MATPRODLINHA I ,
                                         MATPPONLINHA I  );
          CONTLINHACLRO:= CONTLINHACLRO + 1;
         END;
 IF POSPONTOITEMCLOS(ITEM) < TAMPRODUCAO(PRODITEMCLOS(ITEM))
  THEN
   BEGIN
    SIMBOL:= SIMBPROD(POSSIMBPONTOITEMCLOS(ITEM));
    IF SIMBOL ETERMINAL
     THEN
      ADDACTIONSLR(SIMBOL,SHIFT,GOTO(CONJITEM,SIMBOL));
   END
  ELSE
      FOR TERMINAL:= PRIMEIROTERMINAL STEP 1 UNTIL
                       ULTIMOTERMINAL DO
        IF FOLLOW(NAOTERMESQUERDO(PRODITEMCLOS(ITEM)),TERMINAL)
         THEN
           ADDACTIONSLR(TERMINAL,REDUCE,
                                    PRODITEMCLOS(ITEM));
END;
 IF RESTO(CONTTER,11)¬= 0
  THEN
   IF INICIONOVOCITEM
    THEN
```

```
    BEGIN
     IF CONTLINHACLRO > 75
      THEN CABECALHOCLRO;
     WRITE(IMPRE1 SPACE 1 );
      WRITE(IMPRE1,<I8,X1,11(I6,X1,I3,"|")>,CONJITEM,
        FOR I:= 0 STEP 1 UNTIL RESTO(CONTTER-1,11) DO
                                    MATPRODLINHA I ,
                                    MATPPONLINHA I  );
      CONTLINHACLRO:= CONTLINHACLRO + 2;
     END
    ELSE
     BEGIN
      IF CONTLINHACLRO > 75
       THEN CABECALHOCLRO;
      WRITE(IMPRE1,<X9 ,11(I6,X1,I3,"|")>,
        FOR I:= 0 STEP 1 UNTIL RESTO(CONTTER-1,11) DO
                                    MATPRODLINHA I ,
                                    MATPPONLINHA I  );
      CONTLINHACLRO:= CONTLINHACLRO + 1;
     END;
IF RESTO(CONTTERPARSER,5)¬= 0
 THEN
  IF INICIONOVOCITEMPARSER
   THEN
    BEGIN
     IF CONTLINHAPSLR > 75
      THEN CABECALHOPSLR;
     WRITE(IMPRE2 SPACE 1 );
      WRITE(IMPRE2,<I8,X1,5(I6,X1,A6,X1,I8,"|")>,
       CONJITEM,FOR I:= 0 STEP 1 UNTIL RESTO(CONTTERPARSER-1,5) DO
                                        MATTERMLINHA I ,
                                   NACT MATACAOLINHA I  ,
                                        MATCAUXLINHA I  );
      CONTLINHAPSLR:= CONTLINHAPSLR + 2;
     END
    ELSE
     BEGIN
      IF CONTLINHAPSLR > 75
       THEN CABECALHOPSLR;
      WRITE(IMPRE2,<X9,5(I6,X1,A6,X1,I8,"|")>,
```

```
            FOR I:= 0 STEP 1 UNTIL RESTC(CONTTERPARSER-1,5) DO
                                        MATTERMLINHA I ,
                                    NACT MATACAOLINHA I   ,
                                        MATCAUXLINHA I   );
             CONTLINHAPSLR:= CONTLINHAPSLR + 1;
           END;
  END;
END GERACTICNSLR;
%                          ************************
%*************************************************************************
% PROCESSAMENTO LALR(1)                                                  *
%*************************************************************************
PROCEDURE LALR;
BEGIN
 INTEGER ARRAY
   MATITEMLALR1 1:SIZEITEMLALR1 ,
   MATCONJITEMLALR1 0:SIZECITEM ,
   MATPROXITEMLALR1 1:SIZEITEMLALR1 ;
 DEFINE
   PRODITEMLALR1(PITEM)=MATITEMLALR1 PITEM . 47:16 #,
   POSPONTOITEMLALR1(PITEM)=MATITEMLALR1 PITEM . 31:16 #,
   LOOKAHEADITEMLALR1(PITEM)=MATITEMLALR1 PITEM . 15:16 #,
   POSPRIMITEMLALR1(CITEM)=MATCONJITEMLALR1 CITEM . 47:16 #,
    POSULTITEMLALR1(CITEM)=MATCONJITEMLALR1 CITEM . 31:16 #,
   POSPROXITEMLALR1(ITEM)=MATPROXITEMLALR1 ITEM #;
%*************************************************************************
% ROTINA DE GERACAO DOS CONJUNTOS DE ITENS LALR(1)                       *
%*************************************************************************
PROCEDURE GERCONJITEMLALR1;
BEGIN
 INTEGER
   POSINSITEMPROPAG,
   POSPROXITEM,
   NACTERMINAL,
   POSPONTC,
  NITEM,
  PRODUCAC,
   LOCKAHEAD,
   TERMINAL,
   CONJITEM,
```

```
    POSITEM,
    ITEM,
    SIMBOLC,
    POSINSITEM,
    TOPO;
  INTEGER
   ARRAY
    MATITEMLROPROPAG 1:SIZEITEMLALR1 ,
    MATPROPAG 0:NUMTPROD,0:TAMMAXPROD ,
    LR1STKCITEM 1:SIZEITEMLALR1 ,
    LR1STKITEM 1:SIZEITEMLALR1 ;
  DEFINE
    SIMBGCTO(POSITEM)=MATLADODIREITOPROD MATPRODUCAO PROCITEMPROPAG
                  (POSITEM) . 47:16  + POSPONTOITEMPROPAG(POSITEM)-1 #,
    POSPRIMITEMPROPAG(PROD,POSP)=MATPROPAG PROD,POSP . 47:16 #,
    POSULTITEMPROPAG(PROD,POSP)=MATPROPAG PROD,POSP . 31:16 #,
    POSPROXITEMPROPAG(PITEM)=MATITEMLROPROPAG PITEM . 15:16 #,
    PRODITEMPROPAG(PITEM)=MATITEMLROPROPAG PITEM . 47:16 #,
    POSPONTOITEMPROPAG(PITEM)=MATITEMLROPROPAG PITEM . 31:16 #,
    PILHAVAZIA=TOPO ¬= 0#,
    DUMMYLOCKAHEAD=65535#,
    PRODITEMPILHA(NITEM)=LR1STKITEM NITEM . 47:16 #,
    POSPONTOITEMPILHA(NITEM)=LR1STKITEM NITEM . 31:16 #,
    LOOKAHEADITEMPILHA(NITEM)=LR1STKITEM NITEM . 15:16 #,
    CONJITEMITEMPILHA(NITEM)=LR1STKCITEM NITEM #;
PROCEDURE CLOSURELR1ITEM(PRODITEM,POSPONTOITEM);
INTEGER PRODITEM,POSPONTOITEM;
BEGIN
 INTEGER
  PANA,
  SIMBOL,
  SIMBDIR,
  POSSIMBOL,
  TERMINAL,
  PRODUCAO;
 BOOLEAN
   TERMINOU;
 BOOLEAN
  ARRAY
    ADDDLA 0:ENTIER((NUMNAOTERM-1)/48) ,
```

```
  MATADD 1:NUMNAOTERM,0:INDMAX ;
 DEFINE
  ADDCLOSDUMMYLOOKAHEAD(SIMB)=ADDDLA ENTIER((SIMB-1)/48) .
                              (SIMB-1 - 48*ENTIER((SIMB-1)/48)):1 #,
  ADD(SIMB,TERM)=MATADD SIMB,ENTIER((TERM-NUMNAOTERM-1)/48) .
                (TERM-NUMNAOTERM-1 - 48*ENTIER((TERM-NUMNAOTERM
                                        -1)/48)):1 #;
PROCEDURE ADDCLOS(NAOTERM,TERM);
INTEGER NAOTERM,TERM;
BEGIN
 INTEGER
  PRODUCAO;
 IF TERM -= DUMMYLOOKAHEAD
  THEN
   BEGIN
   IF ¬ADD(NAOTERM,TERM)
    THEN
     FOR PRODUCAO:= PRIMEIRAPRODUCAO(NAOTERM) STEP 1 UNTIL
                    ULTIMAPRODUCAO(NAOTERM) DO
      BEGIN
      ADD(NAOTERM,TERM):= TRUE;
       NUMITEMCLOS:= NUMITEMCLOS + 1;
       PRODITEMCLOS(NUMITEMCLOS):= PRODUCAO;
       POSPONTOITEMCLOS(NUMITEMCLOS):= 0;
       LOOKAHEADITEMCLOS(NUMITEMCLOS):= TERM;
      END;
   END
  ELSE
   IF ¬ADDCLOSDUMMYLOOKAHEAD(SIMBOL)
    THEN
         BEGIN
          ADDCLOSDUMMYLOOKAHEAD(SIMBOL):= TRUE;
          FOR PRODUCAO:= PRIMEIRAPRODUCAO(SIMBOL) STEP 1 UNTIL
                         ULTIMAPRODUCAO(SIMBOL) DO
           BEGIN
            NUMITEMCLOS:= NUMITEMCLOS + 1;
            PRODITEMCLOS(NUMITEMCLOS):= PRODUCAO;
            POSPONTOITEMCLOS(NUMITEMCLOS):= 0;
            LOOKAHEADITEMCLOS(NUMITEMCLOS):= DUMMYLOOKAHEAD;
           END;
```

```
          END;
END ADDCLOS;
 NUMITEMCLOS:= 1;
 PRODITEMCLOS(NUMITEMCLOS):= PRODITEM;
 POSPONTOITEMCLOS(NUMITEMCLOS):= POSPONTOITEM;
 LOOKAHEADITEMCLOS(NUMITEMCLOS):= DUMMYLOOKAHEAD;
 PANA:= 1;
 WHILE PANA <= NUMITEMCLOS DO
  BEGIN
   IF POSPONTOITEMCLOS(PANA) < TAMPRODUCAO(PRODITEMCLOS(PANA))
    THEN
     BEGIN
     SIMBOL:= SIMBPROD(POSSIMBPONTOITEMCLOS(PANA));
     IF SIMBOL ENAOTERMINAL
      THEN
       BEGIN
        TERMINOU:= FALSE;
        FOR POSSIMBOL:= POSPRIMSIMBOL(PRODITEMCLOS(PANA)) +
                              POSPONTOITEMCLOS(PANA) + 1 STEP 1
                    WHILE POSSIMBOL <= POSULTSIMBOL(PRODITEMCLOS(PANA))
                              AND ¬TERMINOU DO
         BEGIN
          SIMBDIR:= SIMBPROD(POSSIMBOL);
          IF SIMBDIR ENAOTERMINAL
           THEN
            BEGIN
             FOR TERMINAL:= PRIMEIROTERMINAL STEP 1 UNTIL
                          ULTIMOTERMINAL DO
              IF FIRST(SIMBDIR,TERMINAL)
               THEN
                ADDCLOS(SIMBOL,TERMINAL);
             IF ¬GERAEPSILON(SIMBDIR)
              THEN TERMINOU:= TRUE;
            END
           ELSE
            BEGIN
             ADDCLOS(SIMBOL,SIMBDIR);
             TERMINOU:= TRUE;
            END;
         END;
```

```
          IF ¬TERMINOU
           THEN
             ADDCLOS(SIMBOL,LOOKAHEADITEMCLOS(PANA));
         END;
       END;
    PANA:= PANA + 1;
   END;
END CLOSURELR1ITEM;
PROCEDURE INSERT(CITEM,PROD,POSPONTC,LAHEAD);
INTEGER CITEM,PROD,POSPONTO,LAHEAD;
BEGIN
 INTEGER
   PROXITEM,
   ITEMANT,
   ITEM;
 BOOLEAN
   ON,
   TERMINOU;
 ON:= FALSE;
 TERMINOU:= FALSE;
 ITEM. 47:16 := PROD;
 ITEM. 31:16 := POSPONTO;
 ITEM. 15:16 := LAHEAD;
 ITEMANT:= 0;
 PROXITEM:= POSPRIMITEMLALR1 CITEM ;
 WHILE PROXITEM ¬= TERRA AND ¬TERMINOU DO
  BEGIN
   IF MATITEMLALR1 PROXITEM . 47:48  > ITEM. 47:48
    THEN
     TERMINOU:= TRUE
    ELSE
     IF ITEM. 47:48  = MATITEMLALR1 PROXITEM . 47:48
      THEN
       BEGIN
        TERMINOU:= TRUE;
        ON:= TRUE;
       END
      ELSE
       BEGIN
        ITEMANT:= PROXITEM;
```

```
          PROXITEM:= POSPROXITEMLALR1(PROXITEM);
        END;
  END;
 IF ¬ON
  THEN
   BEGIN
    TOPO:= TOPO + 1;
    CONJITEMITEMPILHA(TOPO):= CITEM;
    PRODITEMPILHA(TOPO):= PROD;
    POSPONTOITEMPILHA(TOPO):= POSPONTO;
    LOOKAHEADITEMPILHA(TOPO):= LAHEAD;
    IF POSPRIMITEMLALR1(CITEM) = TERRA
     THEN
      POSPRIMITEMLALR1(CITEM):= POSINSITEM
     ELSE
     IF ITEMANT = 0
      THEN
       BEGIN
        POSPRIMITEMLALR1(CITEM):= POSINSITEM;
        POSPROXITEMLALR1(POSINSITEM):= PROXITEM;
       END
      ELSE
       BEGIN
        POSPROXITEMLALR1(ITEMANT):= POSINSITEM;
        POSPROXITEMLALR1(POSINSITEM):= PROXITEM;
       END;
    PRODITEMLALR1(POSINSITEM):= PROD;
    POSPONTOITEMLALR1(POSINSITEM):= POSPONTO;
    LOOKAHEADITEMLALR1(POSINSITEM):= LAHEAD;
    POSINSITEM:= POSINSITEM + 1;
   END;
END INSERT;
PROCEDURE PROPAGA(PROD1,POSPONTO1,PROD2,POSPONTO2);
INTEGER PROD1,POSPONTO1,PROD2,POSPONTO2;
BEGIN;
 IF POSPRIMITEMPROPAG(PROD1,POSPONTO1) = TERRA
  THEN
   POSPRIMITEMPROPAG(PROD1,POSPONTO1):= POSINSITEMPROPAG
  ELSE
   POSPROXITEMPROPAG(POSULTITEMPROPAG(PROD1,POSPONTO1)):=
```

```
                                        POSINSITEMPROPAG;
 POSULTITEMPROPAG(PROD1,POSPONTO1):= POSINSITEMPROPAG;
 POSPROXITEMPROPAG(POSINSITEMPROPAG):= TERRA;
 PRODITEMPROPAG(POSINSITEMPROPAG):= PROD2;
 POSPONTOITEMPROPAG(POSINSITEMPROPAG):= POSPONTO2;
 POSINSITEMPROPAG:= POSINSITEMPROPAG + 1;
END PROPAGA;
 POSINSITEM:= 1;
 POSINSITEMPROPAG:= 1;
 INSERT(C,C,O,DOLLAR);
 FOR CONJITEM:= PRIMCONJITEM STEP 1 UNTIL ULTCONJITEM DO
  FOR POSITEM:= POSPRIMITEMLRO(CONJITEM) STEP 1
                                UNTIL POSULTITEMLRO(CONJITEM) DO
   BEGIN
    CLOSURELR1ITEM(PRODITEMLRO(POSITEM),POSPONTOITEMLRO(POSITEM));
    FOR ITEM:= 1 STEP 1 UNTIL NUMITEMCLOS DO
    IF TAMPRODUCAO(PRODITEMCLOS(ITEM)) > POSPONTOITEMCLOS(ITEM)
     THEN
     IF LOOKAHEADITEMCLOS(ITEM) ¬= DUMMYLOOKAHEAD
      THEN
        INSERT(GOTO(CONJITEM,SIMBPROD(POSSIMBPONTOITEMCLOS(ITEM))),
              PRODITEMCLOS(ITEM),
              POSPONTOITEMCLOS(ITEM)+1,LOOKAHEADITEMCLOS(ITEM))
      ELSE
        PROPAGA(PRODITEMLRO(POSITEM),POSPONTOITEMLRO(POSITEM),
                PRODITEMCLOS(ITEM),POSPONTOITEMCLOS(ITEM)+1);
   END;
 WHILE PILHANAOVAZIA DO
  BEGIN
   CONJITEM:= CONJITEMITEMPILHA(TOPO);
   PRODUCAO:= PRODITEMPILHA(TOPO);
   POSPONTO:= POSPONTOITEMPILHA(TOPO);
   LOOKAHEAD:= LOOKAHEADITEMPILHA(TOPO);
   POSPROXITEM:= POSPRIMITEMPROPAG(PRODUCAO,POSPONTO);
   TOPO:= TOPO - 1;
   WHILE POSPROXITEM ¬= TERRA DO
    BEGIN
     INSERT(GOTO(CONJITEM,SIMBGOTO(POSPROXITEM)),PRODITEMPROPAG
           (POSPROXITEM),POSPONTOITEMPROPAG(POSPROXITEM),LOOKAHEAD);
     POSPROXITEM:= POSPROXITEMPROPAG(POSPROXITEM);
```

```
    END;
  END;
END GERCONJITEMLALR1;
%                              ***************************
%*****************************************************************************
% ROTINA DE CALCULO DO FECHAMENTO DE UM CONJUNTO DE ITENS LALR(1)      *
%*****************************************************************************
PROCEDURE CLOSURELALR1(CITEM);
INTEGER CITEM;
BEGIN
 INTEGER
   PANA,
   POSITEM,
   SIMBOL,
   POSSIMBOL,
   SIMBDIR,
   TERMINAL,
   PRODUCAC;
 BOOLEAN
   TERMINCU;
 BOOLEAN
  ARRAY
   MATADD 1:NUMNAOTERM,0:INDMAX ;
 DEFINE
   ADD(SIMB,TERM)=MATADD SIMB,ENTIER((TERM-NUMNAOTERM-1)/48) .
                  (TERM-NUMNAOTERM-1 - 48*ENTIER((TERM-NUMNAOTERM
                                                  -1)/48)):1 #;
PROCEDURE ADDCLOS(NAOTERM,TERM);
INTEGER NAOTERM,TERM;
BEGIN
 INTEGER
   PRODUCAC;
 IF -ADD(NAOTERM,TERM)
  THEN
   FOR PRODUCAO:= PRIMEIRAPRODUCAC(NAOTERM) STEP 1 UNTIL
                   ULTIMAPRODUCAC(NAOTERM) DO
    BEGIN
     ADD(NAOTERM,TERM):= TRUE;
     NUMITEMCLOS:= NUMITEMCLOS + 1;
     PRODITEMCLOS(NUMITEMCLOS):= PRODUCAO;
```

```
      POSPONTOITEMCLOS(NUMITEMCLOS):= 0;
      LOOKAHEADITEMCLOS(NUMITEMCLOS):= TERM;
     END;
END ADDCLOS;
 NUMITEMCLOS:= 0;
 POSITEM:= POSPRIMITEMLALR1(CITEM);
 WHILE POSITEM ¬= TERRA DO
  BEGIN
   NUMITEMCLOS:= NUMITEMCLOS + 1;
   PRODITEMCLOS(NUMITEMCLOS):= PRODITEMLALR1(POSITEM);
   POSPONTOITEMCLOS(NUMITEMCLOS):= POSPONTOITEMLALR1(POSITEM);
   LOOKAHEADITEMCLOS(NUMITEMCLOS):= LOOKAHEADITEMLALR1(POSITEM);
   POSITEM:= POSPROXITEMLALR1(POSITEM);
  END;
 PANA:= 1;
 WHILE PANA <= NUMITEMCLOS DO
  BEGIN
   IF POSPONTOITEMCLOS(PANA) < TAMPRODUCAO(PRODITEMCLOS(PANA))
    THEN
     BEGIN
     SIMBOL:= SIMBPROD(POSSIMBPONTOITEMCLOS(PANA));
     IF SIMBOL ENAOTERMINAL
      THEN
       BEGIN
        TERMINOU:= FALSE;
        FOR POSSIMBOL:= POSPRIMSIMBOL(PRODITEMCLOS(PANA)) +
                              POSPONTOITEMCLOS(PANA) + 1 STEP 1
                    WHILE POSSIMBOL <= POSULTSIMBOL(PRODITEMCLOS(PANA))
                              AND ¬TERMINOU DO
         BEGIN
          SIMBDIR:= SIMBPROD(POSSIMBOL);
          IF SIMBDIR ENAOTERMINAL
           THEN
            BEGIN
             FOR TERMINAL:= PRIMEIROTERMINAL STEP 1 UNTIL
                          ULTIMOTERMINAL DO
              IF FIRST(SIMBDIR,TERMINAL)
               THEN
                ADDCLOS(SIMBOL,TERMINAL);
             IF ¬GERAEPSILON(SIMBDIR)
```

```
              THEN TERMINOU:= TRUE;
            END
           ELSE
            BEGIN
             ADDCLOS(SIMBOL,SIMBDIR);
             TERMINOU:= TRUE;
            END;
         END;
        IF ¬TERMINOU
         THEN
          ADDCLOS(SIMBOL,LOOKAHEADITEMCLOS(PANA));
       END;
     END;
   PANA:= PANA + 1;
  END;
END CLOSURELALR1;
%                              ***************************
%*************************************************************************
% ROTINA DE GERACAO DA TABELA DE ANALISE SINTATICA LALR(1)                *
%*************************************************************************
PROCEDURE GERACTIONLALR1;
BEGIN
 INTEGER
   CONJITEM,
   TERMINAL,
   ITEM,
   CONTLINHACONFLITO,
   CONTTER,
   CONTLINHACLALR1,
   I,
   CONTTERPARSER,
   CONTLINHAPLALR1,
   SIMBOL;
 BOOLEAN
   INICIONOVOCITEM,
   INICIONOVOCITEMPARSER;
 INTEGER
  ARRAY
   MATPROCLINHA 0:5 ,
   MATPPONLINHA 0:5 ,
```

```
   MATLAHELINHA 0:5 ,
   MATTERMLINHA 0:4 ,
   MATACACLINHA 0:4 ,
   MATCAUXLINHA 0:4 ,
   ACTIONLALR NUMNAOTERM+1:NUMNAOTERM+NUMTERM ;
 DEFINE
   INICIALIZARLINHATABELALALR1=
   FOR TERMINAL:= PRIMEIROTERMINAL STEP 1 UNTIL ULTIMOTERMINAL DO
    ACTIONLALR TERMINAL := 0#;
PROCEDURE CABECALHOPLALR1;
BEGIN
 WRITE(IMPRE2 SKIP 1 );
 WRITE(IMPRE2,<X56,"GERADOR PARSER LALR1">);
 WRITE(IMPRE2 SPACE 1 ,<X57,"TABELA RESULTADO">);
 WRITE(IMPRE2 SPACE 1 ,<132("-")>);
 WRITE(IMPRE2 SPACE 1 ,<"        |",5("  TER          ESTADO |")>);
 WRITE(IMPRE2 SPACE 1 ,<"  ESTADO|",5(" MINAL  ACAO  PRODUCAO|")>);
 WRITE(IMPRE2 SPACE 1 ,<132("-")>);
 CONTLINHAPLALR1:= 0
END CABECALHOPLALR1;
PROCEDURE CABECALHOCLALR1;
BEGIN
 WRITE(IMPRE1 SKIP 1 );
 WRITE(IMPRE1,<X52,"CONJUNTOS DE ITENS LALR1">);
 WRITE(IMPRE1 SPACE 1 ,<X56,"TABELA RESULTADO">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 WRITE(IMPRE1 SPACE 1 ,<"CONJUNTO|",6("       POS         ")>);
 WRITE(IMPRE1 SPACE 1 ,<"   DE   |",6(" PRODU PON   LOOK|")>);
 WRITE(IMPRE1 SPACE 1 ,<"  ITENS |",6("  CAO   TO  AHEAD|")>);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 CONTLINHACLALR1:= 0;
END CABECALHOCLALR1;
PROCEDURE IMPCONFLITOLALR1(TERM,TIPO,NUM);
INTEGER TERM,TIPO,NUM;
BEGIN
PROCEDURE CABECALHO;
BEGIN
 WRITE(IMPRE3 SKIP 1 );
 WRITE(IMPRE3,<X26,"GERADOR PARSER LALR">);
 WRITE(IMPRE3 SPACE 1 ,<X31,"CONFLITOS">);
```

```
 WRITE(IMPRE3 SPACE 1 ,<70("-")>);
 WRITE(IMPRE3 SPACE 1 ,<X24,18("-"),X4,18("-")>);
 WRITE(IMPRE3 SPACE 1 ,<X4,"ESTADO",X3,"TERMINAL",2(X14,"ESTADO",X2)>);
 WRITE(IMPRE3 SPACE 1 ,<X24,2(" ACAO      PRODUCAO    ")>);
 WRITE(IMPRE3 SPACE 1 ,<70("-")>);
 CONTLINHACONFLITO:= 0;
END CABECALHO;
 IF CONTLINHACONFLITO > 75
  THEN CABECALHO;
 WRITE(IMPRE3 SPACE 1 );
 WRITE(IMPRE3,<X2,I8,X4,I6,X4,A6,X4,I8,X4,A6,X4,I8>,
         CONJITEM,TERM,NACT TIPO ,NUM,NACT ACTIONLALR TERM . 47:2  ,
          ACTIONLALR TERM . 45:46 );
 CONTLINHACONFLITO:= CONTLINHACONFLITO + 1;
END IMPCONFLITOLALR1;
PROCEDURE ADDACTIONLALR(TERM,TIPO,NUM);
INTEGER TERM,TIPO,NUM ;
BEGIN
 IF ACTIONLALR TERM . 47:2  ¬= 0
  THEN
   BEGIN
   IF ¬(ACTIONLALR TERM . 47:2  = TIPO AND
        ACTIONLALR TERM . 45:46  = NUM )
    THEN
     BEGIN
      IMPCONFLITOLALR1(TERM,TIPO,NUM);
     END
   END
  ELSE
   BEGIN
    ACTIONLALR TERM . 47:2  := TIPO;
    ACTIONLALR TERM . 45:46 := NUM;
    CONTTERPARSER:= CONTTERPARSER + 1;
    MATTERMLINHA RESTO(CONTTERPARSER-1,5) := TERM;
    MATACAOLINHA RESTO(CONTTERPARSER-1,5) := TIPO;
    MATCAUXLINHA RESTO(CONTTERPARSER-1,5) := NUM;
    IF RESTO(CONTTERPARSER,5) = 0
     THEN
      IF INICIONOVOCITEMPARSER
       THEN
```

```
        BEGIN
         IF CONTLINHAPLALR1 > 75
          THEN CABECALHOPLALR1;
         WRITE(IMPRE2 SPACE 1 );
         WRITE(IMPRE2,<I8,X1,5(I6,X1,A6,X1,I8,"|")>,
           CONJITEM,FOR I:= 0 STEP 1 UNTIL 4 DO MATTERMLINHA I  ,
                                              NACT MATACAOLINHA I  ,
                                                   MATCAUXLINHA I  );
         INICIONOVOCITEMPARSER:= FALSE;
         CONTLINHAPLALR1:= CONTLINHAPLALR1 + 2;
        END
       ELSE
         BEGIN
          IF CONTLINHAPLALR1 >75
           THEN CABECALHOPLALR1;
          WRITE(IMPRE2,<X9,5(I6,X1,A6,X1,I8,"|")>,
            FOR I:= 0 STEP 1 UNTIL 4 DO MATTERMLINHA I  ,
                                     NACT MATACAOLINHA I  ,
                                          MATCAUXLINHA I  );
          CONTLINHAPLALR1:= CONTLINHAPLALR1 + 1;
         END;
  END;
END ADDACTIONLALR;
 CABECALHOCLALR1;
 CABECALHOPLALR1;
 CONTLINHACONFLITO:= 76;
 FOR CONJITEM:= PRIMCONJITEM STEP 1 UNTIL ULTCONJITEM DO
  BEGIN
   INICIONOVOCITEM:= TRUE;
   CONTTER:= 0;
   INICIONOVOCITEMPARSER:= TRUE;
   CONTTERPARSER:= 0;
   INICIALIZARLINHATABELALALR1;
   FOR TERMINAL:= PRIMEIROTERMINAL STEP 1 UNTIL ULTIMOTERMINAL DO
    IF GOTO(CONJITEM,TERMINAL) ¬= 0
     THEN
      ADDACTIONLALR(TERMINAL,SHIFT,GOTO(CONJITEM,TERMINAL));
   CLOSURELALR1(CONJITEM);
   FOR ITEM:= 1 STEP 1 UNTIL NUMITEMCLOS DO
       BEGIN
```

```
    CONTTER:= CONTTER + 1 ;
    MATPRODLINHA RESTO(CONTTER-1,6) := PRODITEMCLOS(ITEM);
    MATPPONLINHA RESTO(CONTTER-1,6) := PCSPONTOITEMCLOS(ITEM);
    MATLAHELINHA RESTO(CONTTER-1,6) := LOOKAHEADITEMCLOS(ITEM);
    IF RESTO(CONTTER,6) = 0
     THEN
      IF INICIONOVOCITEM
       THEN
        BEGIN
         IF CONTLINHACLALR1 > 75
          THEN CABECALHOCLALR1;
         WRITE(IMPRE1 SPACE 1 );
         WRITE(IMPRE1,<I8,X1,6(I6,X1,I3,X1,I6,"|")>,
           CONJITEM,FOR I:= 0 STEP 1 UNTIL 5 DO MATPRODLINHA I ,
                                                 MATPPONLINHA I ,
                                                  MATLAHELINHA I  );
         INICIONOVOCITEM:= FALSE;
         CONTLINHACLALR1 := CONTLINHACLALR1 + 2;
        END
       ELSE
        BEGIN
         IF CONTLINHACLALR1 > 75
          THEN CABECALHOCLALR1;
         WRITE(IMPRE1,<X9, 6(I6,X1,I3,X1,I6,"|")>,
           FOR I:= 0 STEP 1 UNTIL 5 DO MATPRODLINHA I ,
                                       MATPPONLINHA I ,
                                       MATLAHELINHA I  );
         CONTLINHACLALR1:= CONTLINHACLALR1 + 1;
        END;
 IF PCSPONTOITEMCLOS(ITEM) = TAMPRODUCAO(PRODITEMCLOS(ITEM))
  THEN
      ADDACTIONLALR(LOOKAHEADITEMCLOS(ITEM),
                    REDUCE,PRODITEMCLOS(ITEM));
END;
 IF RESTO(CONTTER,6)¬= 0
  THEN
   IF INICIONOVOCITEM
    THEN
     BEGIN
      IF CONTLINHACLALR1 > 75
```

```
      THEN CABECALHOCLALR1;
     WRITE(IMPRE1 SPACE 1 );
      WRITE(IMPRE1,<I8,X1,6(I6,X1,I3,X1,I6,"|")>,CONJITEM,
        FOR I:= 0 STEP 1 UNTIL RESTO(CONTTER-1,6) DO
                                    MATPRODLINHA I ,
                                    MATPPONLINHA I ,
                                    MATLAHELINHA I  );
      CONTLINHACLALR1:= CONTLINHACLALR1 + 2;
     END
    ELSE
     BEGIN
      IF CONTLINHACLALR1 > 75
       THEN CABECALHOCLALR1;
      WRITE(IMPRE1,<X9 ,6(I6,X1,I3,X1,I6,"|")>,
        FOR I:= 0 STEP 1 UNTIL RESTO(CONTTER-1,6) DO
                                    MATPRODLINHA I ,
                                    MATPPONLINHA I ,
                                    MATLAHELINHA I  );
      CONTLINHACLALR1:= CONTLINHACLALR1 + 1;
     END;
IF RESTO(CONTTERPARSER,5)¬= 0
 THEN
  IF INICIONOVOCITEMPARSER
   THEN
    BEGIN
     IF CONTLINHAPLALR1 > 75
      THEN CABECALHOPLALR1;
     WRITE(IMPRE2 SPACE 1 );
      WRITE(IMPRE2,<I8,X1,5(I6,X1,A6,X1,I8,"|")>,
       CONJITEM,FOR I:= 0 STEP 1 UNTIL RESTO(CONTTERPARSER-1,5) DO
                                        MATTERMLINHA I ,
                                   NACT MATACAOLINHA I  ,
                                        MATCAUXLINHA I  );
      CONTLINHAPLALR1:= CONTLINHAPLALR1 + 2;
     END
    ELSE
     BEGIN
      IF CONTLINHAPLALR1 > 75
       THEN CABECALHOPLALR1;
      WRITE(IMPRE2,<X9,5(I6,X1,A6,X1,I8,"|")>,
```

```
          FOR I:= 0 STEP 1 UNTIL RESTO(CONTTERPARSER-1,5) DO
                                    MATTERMLINHA I ,
                                NACT MATACAOLINHA I   ,
                                    MATCAUXLINHA I   );
           CONTLINHAPLALR1:= CONTLINHAPLALR1 + 1;
          END;
  END;
END GERACTIONLALR1;
%                          ************************
 GERCONJITEMLALR1;
 GERACTIONLALR1;
END LALR;
%                          ************************
 GERCONJITEMLRO;
 IMPGOTOLRO;
 IF PROC(9)
  THEN GERACTIONSLR;
 IF PROC(7)
  THEN LALR;
END SLRLALR;
%                          ************************
%*************************************************************************
 % PROCESSAMENTO LR(1)                                                   *
%*************************************************************************
PROCEDURE LR1;
BEGIN
 INTEGER
  ARRAY
   MATCONJITEM 0:SIZECITEMLR1 ,
   MATITEM 1:SIZEITEMLR1 ,
   MATGOTO 0:SIZECITEMLR1,1:NUMNAOTERM+NUMTERM ;
 INTEGER
   CITEMMAX;
 DEFINE
   POSPRIMITEMLR1(CITEM)=MATCONJITEM CITEM . 47:16 #,
    POSULTITEMLR1(CITEM)=MATCONJITEM CITEM . 47:16  +
                         MATCONJITEM CITEM . 31:16  - 1#,
   PRODITEMLR1(PITEM)=MATITEM PITEM . 47:16 #,
   POSPONTOITEMLR1(PITEM)=MATITEM PITEM . 31:16 #,
   LOOKAHEADITEMLR1(PITEM)=MATITEM PITEM . 15:16 #,
```

```
   TAMCONJITEM(CITEM)=MATCONJITEM CITEM . 31:16 #,
   PRIMCONJITEM=0#,
   ULTCONJITEM=CITEMMAX#,
   GOTO(CITEM,SIMBOL)=MATGOTO CITEM,SIMBOL #;
%*************************************************************************
% ROTINA DE CALCULO DO FECHAMENTO DE UM CONJUNTO DE ITENS LR(1)          *
%*************************************************************************
PROCEDURE CLOSURELR1(CITEM);
INTEGER CITEM;
BEGIN
 INTEGER
   PANA,
   POSITEM,
   SIMBOL,
   POSSIMBOL,
   SIMBDIR,
   TERMINAL,
   PRODUCAC;
 BOOLEAN
   TERMINCU;
 BOOLEAN
  ARRAY
   MATADD 1:NUMNAOTERM,0:INDMAX ;
 DEFINE
   ADD(SIMB,TERM)=MATADD SIMB,ENTIER((TERM-NUMNAOTERM-1)/48) .
                  (TERM-NUMNAOTERM-1 - 48*ENTIER((TERM-NUMNAOTERM
                                                  -1)/48)):1 #;
PROCEDURE ADDCLOS(NAOTERM,TERM);
INTEGER NAOTERM,TERM;
BEGIN
 INTEGER
   PRODUCAC;
 IF ¬ADD(NAOTERM,TERM)
  THEN
   FOR PRODUCAO:= PRIMEIRAPRODUCAC(NAOTERM) STEP 1 UNTIL
                  ULTIMAPRODUCAC(NAOTERM) DO
    BEGIN
     ADD(NAOTERM,TERM):= TRUE;
     NUMITEMCLOS:= NUMITEMCLOS + 1;
     PROCITEMCLOS(NUMITEMCLOS):= PRODUCAO;
```

```
      POSPONTOITEMCLOS(NUMITEMCLOS):= 0;
      LOOKAHEADITEMCLOS(NUMITEMCLOS):= TERM;
      PROXITEMCLOS(NUMITEMCLOS-1):= NUMITEMCLOS;
      ITEMCLOSANT(NUMITEMCLOS):= NUMITEMCLOS-1;
     END;
END ADDCLOS;
 PRIMITEMCLOS:= 1;
 NUMITEMCLOS:= 1;
 PRODITEMCLOS(NUMITEMCLOS):= PRODITEMLR1(POSPRIMITEMLR1(CITEM));
 POSPONTOITEMCLOS(NUMITEMCLOS):= POSPONTOITEMLR1(POSPRIMITEMLR1(
                                                        CITEM));
  LOOKAHEADITEMCLOS(NUMITEMCLOS):= LOOKAHEADITEMLR1(
                                   POSPRIMITEMLR1(CITEM));
 ITEMCLOSANT(NUMITEMCLOS):= TERRA;
 FOR POSITEM:= POSPRIMITEMLR1(CITEM) + 1 STEP 1 UNTIL
               POSULTITEMLR1(CITEM) DO
  BEGIN
   NUMITEMCLOS:= NUMITEMCLOS+1;
   PRODITEMCLOS(NUMITEMCLOS):= PRODITEMLR1(POSITEM);
   POSPONTOITEMCLOS(NUMITEMCLOS):= POSPONTOITEMLR1(POSITEM);
   LOOKAHEADITEMCLOS(NUMITEMCLOS):= LOOKAHEADITEMLR1(POSITEM);
   PROXITEMCLOS(NUMITEMCLOS-1):= NUMITEMCLOS;
   ITEMCLOSANT(NUMITEMCLOS):= NUMITEMCLOS - 1;
  END;
 PANA:= 1;
 WHILE PANA <= NUMITEMCLOS DO
  BEGIN
   IF POSPONTOITEMCLOS(PANA) < TAMPRODUCAO(PRODITEMCLOS(PANA))
    THEN
     BEGIN
     SIMBOL:= SIMBPROD(POSSIMBPONTOITEMCLOS(PANA));
     IF SIMBOL ENAOTERMINAL
      THEN
       BEGIN
        TERMINOU:= FALSE;
        FOR POSSIMBOL:= POSPRIMSIMBOL(PRODITEMCLOS(PANA)) +
                        POSPONTOITEMCLOS(PANA) + 1 STEP 1
                 WHILE POSSIMBOL <= POSULTSIMBOL(PRODITEMCLOS(PANA))
                        AND ¬TERMINOU DO
         BEGIN
```

```
          SIMBDIR:= SIMBPROD(POSSIMBOL);
          IF SIMBDIR ENAOTERMINAL
           THEN
            BEGIN
             FOR TERMINAL:= PRIMEIROTERMINAL STEP 1 UNTIL
                          ULTIMOTERMINAL DO
              IF FIRST(SIMBDIR,TERMINAL)
               THEN
                ADDCLOS(SIMBOL,TERMINAL);
             IF ¬GERAEPSILON(SIMBDIR)
              THEN TERMINOU:= TRUE;
            END
           ELSE
            BEGIN
             ADDCLOS(SIMBOL,SIMBDIR);
             TERMINOU:= TRUE;
            END;
         END;
        IF ¬TERMINOU
         THEN
          ADDCLOS(SIMBOL,LOOKAHEADITEMCLOS(PANA));
       END;
     END;
   PANA:= PANA + 1;
  END;
    PROXITEMCLOS(NUMITEMCLOS):= TERRA;
END CLOSURELR1;
%*****************************************************************
% ROTINA DE GERACAO DOS CONJUNTOS DE ITENS LR(1)                 *
%*****************************************************************
PROCEDURE GERCONJITEMLR1;
BEGIN
 INTEGER
   CITEMAN;
%*****************************************************************
% ROTINA DE CALCULO DOS CONJUNTOS DE ITENS POSSIVEIS NO GOTO DO  *
% CONJUNTO DE ITEM CITEM                                         *
%*****************************************************************
PROCEDURE GERAGOTO(CITEM);
INTEGER CITEM;
```

```
BEGIN
 INTEGER
   SIMBOL,
   NITEM,
   ITEM,
   ITEMANT,
   CITEMENCONTRADO;
 BOOLEAN
   ENCONTROU;
 DEFINE
   POSINSPRIMITEM=POSULTITEMLR1(CITEMMAX-1) + 1#,
   POSINSITEM=POSULTITEMLR1(CITEMMAX) + NITEM + 1#;
PROCEDURE PROCURAITEMIDENTICO(ENCONTROU,CITEMID);
BOOLEAN ENCONTROU;
INTEGER CITEMID;
BEGIN
 INTEGER
   POS1,
   IND1,
   IND2,
   INTAUX,
   CONJITEM;
 POS1:= POSULTITEMLR1(CITEMMAX);
 FOR IND1:= POS1+NITEM-1 STEP -1 UNTIL POS1+1 DO
  FOR IND2:= POS1+1 STEP 1 UNTIL IND1 DO
   IF MATITEM IND2+1 . 47:48  < MATITEM IND2 . 47:48
    THEN
     BEGIN
     INTAUX. 47:48 := MATITEM IND2 . 47:48 ;
     MATITEM IND2 . 47:48 := MATITEM IND2+1 . 47:48 ;
     MATITEM IND2+1 . 47:48 := INTAUX. 47:48 ;
     END;
 ENCONTROU:= FALSE;
 FOR CONJITEM:= 1 STEP 1 WHILE CONJITEM <= CITEMMAX AND
                                  ¬ ENCONTROU DO
  BEGIN
   IF TAMCONJITEM(CONJITEM) = NITEM
     THEN
      BEGIN
       FOR IND1:= 0 STEP 1 WHILE MATITEM POSPRIMITEMLR1(CONJITEM)+IND1
```

```
      . 47:48 = MATITEM POS1+1+IND1 . 47:48  AND IND1 < NITEM DO;
        IF IND1 = NITEM
         THEN ENCONTROU:= TRUE;
       END;
  END;
 CITEMID:= CONJITEM - 1;
END PROCURACITEMIDENTICO;
 CLOSURELR1(CITEM);
 WHILE PRIMITEMCLOS ¬= TERRA DO
  BEGIN
   NITEM:= 0;
   IF TAMPRODUCAO(PRODITEMCLOS(PRIMITEMCLOS)) >
      POSPONTOITEMCLOS(PRIMITEMCLOS)
    THEN
     BEGIN
      SIMBOL:= SIMBPROD(POSSIMBPONTOITEMCLOS(PRIMITEMCLOS));
      PRODITEMLR1(POSINSITEM):= PRODITEMCLOS(PRIMITEMCLOS);
      POSPONTOITEMLR1(POSINSITEM):= POSPONTOITEMCLOS(PRIMITEMCLOS)+1;
      LOOKAHEADITEMLR1(POSINSITEM):= LOOKAHEADITEMCLOS(PRIMITEMCLOS);
      NITEM:= NITEM + 1;
      ELIMINACLOS(PRIMITEMCLOS);
      ITEM:= PRIMITEMCLOS;
      WHILE ITEM ¬= TERRA DO
       BEGIN
        IF TAMPRODUCAO(PRODITEMCLOS(ITEM)) >
           POSPONTOITEMCLOS(ITEM)
         THEN
          BEGIN
          IF SIMBPROD(POSSIMBPONTOITEMCLOS(ITEM)) = SIMBOL
           THEN
            BEGIN
             PRODITEMLR1(POSINSITEM):= PRODITEMCLOS(ITEM);
             POSPONTOITEMLR1(POSINSITEM):= POSPONTOITEMCLOS(ITEM) + 1;
              LOOKAHEADITEMLR1(POSINSITEM):= LOOKAHEADITEMCLOS(ITEM);
             NITEM:= NITEM + 1;
             ELIMINACLOS(ITEM);
            END;
          END
         ELSE
           ELIMINACLOS(ITEM);
```

```
        ITEM:= PROXITEMCLOS(ITEM);
       END;
     IF NITEM > 0
      THEN
       BEGIN
        PROCURACITEMIDENTICO(ENCONTROU,CITEMENCONTRADO);
        IF ¬ ENCONTROU
         THEN
          BEGIN
           CITEMMAX:= CITEMMAX + 1;
           POSPRIMITEMLR1(CITEMMAX):= POSINSPRIMITEM;
           TAMCONJITEM(CITEMMAX):= NITEM;
           GOTO(CITEM,SIMBOL):= CITEMMAX;
          END
         ELSE
          GOTO(CITEM,SIMBOL):= CITEMENCONTRADO;
       END;
      END
     ELSE
       ELIMINACLOS(PRIMITEMCLOS);
   END;
END GERAGOTO;
%                              ***************************
 POSPRIMITEMLR1(0):= 1;
 TAMCONJITEM(0):= 1;
 PRODITEMLR1(1):= 0;
 POSPONTOITEMLR1(1):= 0;
 LOOKAHEADITEMLR1(1):= DOLLAR;
 WHILE CITEMAN <= CITEMMAX DO
  BEGIN
   GERAGOTO(CITEMAN);
   CITEMAN:= CITEMAN + 1;
  END;
END GERCONJITEMLR1;
%                              ***************************
%*************************************************************************
% ROTINA DE IMPRESSAO DA RELACAO GOTO PARA CONJUNTOS DE ITENS LR(1)     *
%*************************************************************************
PROCEDURE IMPGOTOLR1;
BEGIN
```

```
 INTEGER
   CONJITEM,
   CONTPAR,
   NAOTERMINAL,
   CONTLINHA,
   I;
 BOOLEAN
   INICIONOVOCITEM;
 INTEGER
  ARRAY
   MATNTERMLINHA 0:6 ,
   MATCITEMLINHA 0:6 ;
PROCEDURE CABECALHO;
BEGIN
 WRITE(IMPRE1 SKIP 1 );
 WRITE(IMPRE1,<X59,"GOTO (LR1)">);
 WRITE(IMPRE1 SPACE 1 ,<X56, "TABELA RESULTADO">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 WRITE(IMPRE1 SPACE 1 ,<"CONJUNTO|", 7("  NAC  CONJUNTO ")>);
 WRITE(IMPRE1 SPACE 1 ,<"    DE  |", 7("  TER    ITEM  |")>);
 WRITE(IMPRE1 SPACE 1 ,<"   ITEM |", 7(" MINAL   FINAL |")>);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 CONTLINHA:= 0;
END CABECALHO;
 CABECALHO;
 FOR CONJITEM:= PRIMCONJITEM STEP 1 UNTIL ULTCONJITEM DO
  BEGIN
   INICIONOVOCITEM:= TRUE;
   CONTPAR:= 0;
   FOR NACTERMINAL:= PRIMEIRONACTERMINAL STEP 1
                                        UNTIL ULTIMONACTERMINAL DO
    IF GOTO(CONJITEM,NACTERMINAL) ¬= 0
     THEN
      BEGIN
       CONTPAR:= CONTPAR + 1;
       MATNTERMLINHA RESTO(CONTPAR-1,7) := NACTERMINAL;
       MATCITEMLINHA RESTO(CONTPAR-1,7) := GOTO(CONJITEM,NACTERMINAL);
       IF RESTO(CONTPAR,7) = 0
        THEN
         IF INICIONOVOCITEM
```

```
        THEN
         BEGIN
          IF CONTLINHA > 75
           THEN CABECALHO;
           WRITE(IMPRE1 SPACE 1 );
          WRITE(IMPRE1,<I8,X1,7(I6,X1,I8,"|")>,CONJITEM,
             FOR I:= 0 STEP 1 UNTIL 6 DO MATNTERMLINHA I ,
                                          MATCITEMLINHA I  );
          INICIONOVOCITEM:= FALSE;
          CONTLINHA:= CONTLINHA + 2;
         END
        ELSE
         BEGIN
          IF CONTLINHA > 75
           THEN CABECALHO;
          WRITE(IMPRE1,<X9,7(I6,X1,I8,"|")>,
             FOR I:= 0 STEP 1 UNTIL 6 DO MATNTERMLINHA I ,
                                          MATCITEMLINHA I  );
          CONTLINHA:= CONTLINHA + 1;
         END;
   END;
IF RESTO(CONTPAR,7) ¬= 0
 THEN
  IF INICIONOVOCITEM
   THEN
    BEGIN
     IF CONTLINHA > 75
      THEN CABECALHO;
      WRITE(IMPRE1 SPACE 1 );
     WRITE(IMPRE1,<I8,X1,7(I6,X1,I8,"|")>,CONJITEM,
       FOR I:= 0 STEP 1 UNTIL RESTO(CONTPAR-1,7) DO
                                  MATNTERMLINHA I ,
                                  MATCITEMLINHA I  );
     CONTLINHA:= CONTLINHA + 2;
    END
   ELSE
    BEGIN
     IF CONTLINHA > 75
      THEN CABECALHO;
     WRITE(IMPRE1,<X9,7(I6,X1,I8,"|")> ,
```

```
          FOR I:= 0 STEP 1 UNTIL RESTO(CONTPAR-1,7) DO
                                        MATNTERMLINHA I ,
                                        MATCITEMLINHA I  );
         CONTLINHA:= CONTLINHA + 1;
        END;
  END;
END IMPGOTOLR1;
%                              ***************************
%*************************************************************************
% ROTINA DE GERACAO DA TABELA DE ANALISE SINTATICA LR(1)                 *
%*************************************************************************
PROCEDURE GERACTIONLR1;
BEGIN
 INTEGER
   CONJITEM,
   TERMINAL,
   ITEM,
   CONTLINHACONFLITO,
   CONTTER,
   CONTLINHALR1,
   I,
   CONTTERPARSER,
   CONTLINHAPLR1,
   SIMBOL;
 BOOLEAN
   INICIONOVOCITEM,
   INICIONOVOCITEMPARSER;
 INTEGER
  ARRAY
   MATPROLINHA 0:5 ,
   MATPPONLINHA 0:5 ,
   MATLAHELINHA 0:5 ,
   MATTERMLINHA 0:4 ,
   MATACALINHA 0:4 ,
   MATCAUXLINHA 0:4 ,
   ACTIONLR NUMNAOTERM+1:NUMNAOTERM+NUMTERM ;
 DEFINE
   INICIALIZARLINHATABELALR1=
   FOR TERMINAL:= PRIMEIROTERMINAL STEP 1 UNTIL ULTIMOTERMINAL DO
    ACTIONLR TERMINAL := 0#;
```

```
PROCEDURE CABECALHOPLR1;
BEGIN
 WRITE(IMPRE2 SKIP 1 );
 WRITE(IMPRE2,<X56,"GERADOR PARSER LR1">);
 WRITE(IMPRE2 SPACE 1 ,<X57,"TABELA RESULTADO">);
 WRITE(IMPRE2 SPACE 1 ,<132("-")>);
 WRITE(IMPRE2 SPACE 1 ,<"         |",5("  TER           ESTADO |")>);
 WRITE(IMPRE2 SPACE 1 ,<"  ESTADO|",5(" MINAL  ACAO  PRODUCAO|")>);
 WRITE(IMPRE2 SPACE 1 ,<132("-")>);
 CONTLINHAPLR1:= 0
END CABECALHOPLR1;
PROCEDURE CABECALHOCLR1;
BEGIN
 WRITE(IMPRE1 SKIP 1 );
 WRITE(IMPRE1,<X52,"CONJUNTOS DE ITENS LR1">);
 WRITE(IMPRE1 SPACE 1 ,<X56,"TABELA RESULTADO">);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 WRITE(IMPRE1 SPACE 1 ,<"CONJUNTO|",6("       POS        ")>);
 WRITE(IMPRE1 SPACE 1 ,<"    DE   |",6(" PRODU PON   LOOK|")>);
 WRITE(IMPRE1 SPACE 1 ,<"  ITENS |",6("  CAO   TO  AHEAD|")>);
 WRITE(IMPRE1 SPACE 1 ,<132("-")>);
 CONTLINHACLR1:= 0;
END CABECALHOCLR1;
PROCEDURE IMPCONFLITOLR1(TERM,TIPO,NUM);
INTEGER TERM,TIPO,NUM;
BEGIN
PROCEDURE CABECALHO;
BEGIN
 WRITE(IMPRE3 SKIP 1 );
 WRITE(IMPRE3,<X26,"GERADOR PARSER LR1">);
 WRITE(IMPRE3 SPACE 1 ,<X31,"CONFLITOS">);
 WRITE(IMPRE3 SPACE 1 ,<70("-")>);
 WRITE(IMPRE3 SPACE 1 ,<X24,18("-"),X4,18("-")>);
 WRITE(IMPRE3 SPACE 1 ,<X4,"ESTADO",X3,"TERMINAL",2(X14,"ESTADO",X2)>);
 WRITE(IMPRE3 SPACE 1 ,<X24,2(" ACAO     PRODUCAO    ")>);
 WRITE(IMPRE3 SPACE 1 ,<70("-")>);
 CONTLINHACONFLITO:= 0;
END CABECALHO;
 IF CONTLINHACONFLITO > 75
  THEN CABECALHO;
```

```
 WRITE(IMPRE3 SPACE 1 );
 WRITE(IMPRE3,<X2,I8,X4,I6,X4,A6,X4,I8,X4,A6,X4,I8>,
         CONJITEM,TERM,NACT TIPO ,NUM,NACT ACTIONLR TERM . 47:2  ,
          ACTIONLR TERM . 45:46 );
 CONTLINHACONFLITO:= CONTLINHACONFLITO + 1;
END IMPCONFLITOLR1;
PROCEDURE ADDACTIONLR(TERM,TIPO,NUM);
INTEGER TERM,TIPO,NUM ;
BEGIN
 IF ACTIONLR TERM . 47:2  ¬= 0
  THEN
   BEGIN
   IF ¬(ACTIONLR TERM . 47:2  = TIPO AND
        ACTIONLR TERM . 45:46  = NUM )
    THEN
     BEGIN
      IMPCONFLITOLR1(TERM,TIPO,NUM);
     END
   END
  ELSE
   BEGIN
    ACTIONLR TERM . 47:2  := TIPO;
    ACTIONLR TERM . 45:46 := NUM;
    CONTTERPARSER:= CONTTERPARSER + 1;
    MATTERMLINHA RESTO(CONTTERPARSER-1,5) := TERM;
    MATACAOLINHA RESTO(CONTTERPARSER-1,5) := TIPO;
    MATCAUXLINHA RESTO(CONTTERPARSER-1,5) := NUM;
    IF RESTO(CONTTERPARSER,5) = 0
     THEN
      IF INICIONOVOCITEMPARSER
       THEN
        BEGIN
         IF CONTLINHAPLR1 > 75
          THEN CABECALHOPLR1;
         WRITE(IMPRE2 SPACE 1 );
         WRITE(IMPRE2,<I8,X1,5(I6,X1,A6,X1,I8,"|")>,
           CONJITEM,FOR I:= 0 STEP 1 UNTIL 4 DO MATTERMLINHA I ,
                                              NACT MATACAOLINHA I  ,
                                                   MATCAUXLINHA I  );
         INICIONOVOCITEMPARSER:= FALSE;
```

```
          CONTLINHAPLR1:= CONTLINHAPLR1 + 2;
         END
        ELSE
          BEGIN
           IF CONTLINHAPLR1 >75
            THEN CABECALHOPLR1;
           WRITE(IMPRE2,<X9,5(I6,X1,A6,X1,I8,"|")>,
             FOR I:= 0 STEP 1 UNTIL 4 DO MATTERMLINHA I  ,
                                          NACT MATACAOLINHA I  ,
                                               MATCAUXLINHA I  );
           CONTLINHAPLR1:= CONTLINHAPLR1 + 1;
          END;
  END;
END ADDACTIONLR;
 CABECALHOCLR1;
 CABECALHOPLR1;
 CONTLINHACONFLITO:= 76;
 FOR CONJITEM:= PRIMCONJITEM STEP 1 UNTIL ULTCONJITEM DO
  BEGIN
   INICIONOVOCITEM:= TRUE;
   CONTTER:= 0;
   INICIONOVOCITEMPARSER:= TRUE;
   CONTTERPARSER:= 0;
   INICIALIZARLINHATABELALR1;
   FOR TERMINAL:= PRIMEIROTERMINAL STEP 1 UNTIL ULTIMOTERMINAL DO
    IF GOTO(CONJITEM,TERMINAL) ¬= 0
     THEN
      ADDACTIONLR(TERMINAL,SHIFT,GOTO(CONJITEM,TERMINAL));
   CLOSURELR1(CONJITEM);
   FOR ITEM:= 1 STEP 1 UNTIL NUMITEMCLOS DO
       BEGIN
        CONTTER:= CONTTER + 1 ;
        MATPRODLINHA RESTO(CONTTER-1,6) := PRODITEMCLOS(ITEM);
        MATPPONLINHA RESTO(CONTTER-1,6) := POSPONTOITEMCLOS(ITEM);
        MATLAHELINHA RESTO(CONTTER-1,6) := LOOKAHEADITEMCLOS(ITEM);
        IF RESTO(CONTTER,6) = 0
         THEN
          IF INICIONOVOCITEM
           THEN
            BEGIN
```

```
          IF CONTLINHACLR1 > 75
           THEN CABECALHOCLR1;
          WRITE(IMPRE1 SPACE 1 );
          WRITE(IMPRE1,<I8,X1,6(I6,X1,I3,X1,I6,"|")>,
            CONJITEM,FOR I:= 0 STEP 1 UNTIL 5 DO MATPRODLINHA I ,
                                                  MATPPONLINHA I ,
                                                  MATLAHELINHA I  );
          INICIONOVOCITEM:= FALSE;
          CONTLINHACLR1 := CONTLINHACLR1 + 2;
         END
        ELSE
         BEGIN
          IF CONTLINHACLR1 > 75
           THEN CABECALHOCLR1;
          WRITE(IMPRE1,<X9, 6(I6,X1,I3,X1,I6,"|")>,
            FOR I:= 0 STEP 1 UNTIL 5 DO MATPRODLINHA I ,
                                         MATPPONLINHA I ,
                                         MATLAHELINHA I  );
          CONTLINHACLR1:= CONTLINHACLR1 + 1;
         END;
 IF POSPONTOITEMCLOS(ITEM) = TAMPRODUCAO(PRODITEMCLOS(ITEM))
  THEN
      ADDACTIONLR(LOOKAHEADITEMCLOS(ITEM),
                  REDUCE,PRODITEMCLOS(ITEM));
END;
 IF RESTO(CONTTER,6)¬= 0
  THEN
   IF INICIONOVOCITEM
    THEN
     BEGIN
      IF CONTLINHACLR1 > 75
       THEN CABECALHOCLR1;
     WRITE(IMPRE1 SPACE 1 );
      WRITE(IMPRE1,<I8,X1,6(I6,X1,I3,X1,I6,"|")>,CONJITEM,
        FOR I:= 0 STEP 1 UNTIL RESTO(CONTTER-1,6) DO
                                  MATPRODLINHA I ,
                                  MATPPONLINHA I ,
                                  MATLAHELINHA I  );
      CONTLINHACLR1:= CONTLINHACLR1 + 2;
     END
```

```
        ELSE
         BEGIN
          IF CONTLINHACLR1 > 75
           THEN CABECALHOCLR1;
          WRITE(IMPRE1,<X9 ,6(I6,X1,I3,X1,I6,"|")>,
            FOR I:= 0 STEP 1 UNTIL RESTO(CONTTER-1,6) DO
                                    MATPRODLINHA I ,
                                    MATPPONLINHA I ,
                                    MATLAHELINHA I  );
          CONTLINHACLR1:= CONTLINHACLR1 + 1;
         END;
      IF RESTO(CONTTERPARSER,5)¬= 0
       THEN
        IF INICIONOVOCITEMPARSER
         THEN
          BEGIN
           IF CONTLINHAPLR1 > 75
            THEN CABECALHOPLR1;
           WRITE(IMPRE2 SPACE 1 );
           WRITE(IMPRE2,<I8,X1,5(I6,X1,A6,X1,I8,"|")>,
            CONJITEM,FOR I:= 0 STEP 1 UNTIL RESTO(CONTTERPARSER-1,5) DO
                                          MATTERMLINHA I ,
                                    NACT MATACAOLINHA I  ,
                                          MATCAUXLINHA I  );
           CONTLINHAPLR1:= CONTLINHAPLR1 + 2;
          END
         ELSE
          BEGIN
           IF CONTLINHAPLR1 > 75
            THEN CABECALHOPLR1;
           WRITE(IMPRE2,<X9,5(I6,X1,A6,X1,I8,"|")>,
            FOR I:= 0 STEP 1 UNTIL RESTO(CONTTERPARSER-1,5) DO
                                    MATTERMLINHA I ,
                              NACT MATACAOLINHA I  ,
                                    MATCAUXLINHA I  );
           CONTLINHAPLR1:= CONTLINHAPLR1 + 1;
          END;
  END;
END GERACTICNLR1;
%                          ************************
```

```
 GERCONJITEMLR1;
 INPGCTCLR1;
 GERACTICNLR1;
END LR1;
%                              ***************************
 NACT SHIFT := " SHIFT";
 NACT ACCEPT :="ACCEPT";
 NACT REDUCE := "REDUCE";
 IF ¬HOUVEERROGRAVE
  THEN
   BEGIN
    MATSIMBOL 0 . 47:16 := 0;
    MATSIMBOL 0 . 31:16 := 1;
    MATPRODUCAO 0 . 47:16 := 1;
    MATPRODUCAO 0 . 31:16 := 1;
    MATPRODUCAO 0 . 15:16 := 0;
    MATLADODIREITOPROD 1 := 1;
    CONTPROD:= 1;
    CONTLDIR:= 2;
    FOR I:= 1 STEP 1 UNTIL TOTNAOTERM DO
     BEGIN
      PRIMEIRAENTRADA:= TRUE;
      NPROD:= 0;
      POS:= POSPRIMSIMBLDIR(I);
      IF POS ¬= TERRA THEN
      DO
       BEGIN
        TPROD:= 0;
        DO
         BEGIN
          IF PRIMEIRAENTRADA
           THEN PRIMEIRAENTRADA:= FALSE
           ELSE POS:= PROXLDIR(POS);
          IF INDTERM(POS) = 1
           THEN
            MATLADODIREITOPROD CONTLDIR := MATLDIRPROD POS +TOTNAOTERM
           ELSE
            MATLADODIREITOPROD CONTLDIR := MATLDIRPROD POS ;
           TPROD:= TPROD + 1;
            CONTLDIR := CONTLDIR + 1;
```

```
          END
         UNTIL
          PROXLDIR(POS) = TERRA OR FIMPROD(POS) = 1;
         MATPRODUCAO CONTPROD . 47:16 := CONTLDIR  -TPROD;
         MATPRODUCAO CONTPROD . 31:16 := TPROD;
         MATPRODUCAO CONTPROD . 15:16 := I;
         CONTPROD:= CONTPROD + 1;
         NPROD:= NPROD + 1;
        END
       UNTIL
        PROXLDIR(POS) = TERRA;
       IF PRODEPSILON(I) = 1
        THEN
         BEGIN
          MATPRODUCAO CONTPROD . 47:16 := CONTLDIR;
          MATPRODUCAO CONTPROD . 31:16 :=0;
          MATPRODUCAO CONTPROD . 15:16 := I;
          CONTPROD:= CONTPROD + 1;
          NPROD:= NPROD + 1;
         END;
       MATSIMBOL I . 47:16 := CONTPROD -NPROD;
       MATSIMBOL I . 31:16 := NPROD;
     END;
  IMPPRODUCES;
  IF PROC(1)
   THEN GERAFIRST;
  IF PROC(2)
   THEN IMPFIRST;
  IF PROC(3)
   THEN GERAFOLLOW;
  IF PROC(4)
   THEN IMPFOLLOW;
  IF PROC(5)
   THEN SLRLALR;
  IF PROC(11)
   THEN LR1;
     END;
END GERPARSER;
 SHIFT:=1;
 REDUCE:=2;
```

```
ACCEPT:= 3;
IF READ(PARMCONV,/,AREAMAX)
 THEN
  BEGIN
   AREAMAX:= 4000;
  END;
CONVERSCR;
   NUMNACTERM:= TOTNACTERM;
   NUMTERM:= TOTTERM + 1;
   NUMTPROD:= CONTPR;
   NUMTSIMBPROD:= LDIRDISP - 1;
   FOR I:= 1,2              DO MATPROC2 1,I := 1;
   FOR I:= 1,3,4            DO MATPROC2 2,I := 1;
   FOR I:= 1,3,5,6,9        DO MATPROC2 3,I := 1;
   FOR I:= 1,5,7,8,10       DO MATPROC2 4,I := 1;
   FOR I:= 1,11,12,13       DO MATPROC2 5,I := 1;
   READ (PROCED,<5A1>,FOR I:= 1 STEP 1 UNTIL 5 DO MATPROC1 I );
   FOR I:= 1 STEP 1 UNTIL 5 DO
    IF MATPROC1 I  ¬= 4"C000000000"8" "
     THEN
      FOR J:= 1 STEP 1 UNTIL 13 DO
       IF MATPROC2 I,J  = 1
        THEN
         MATPROC3 J := 1;
   INDMAX:= ENTIER((NUMTERM-1)/48);
   IF READ(PARMGERAL,/,SIZESTKCLOS)
    THEN SIZESTKCLOS:= 10000;
   IF PROC(7)
    THEN
     IF READ(PARMLALR,/,SIZEITEMLALR1)
      THEN
       BEGIN
        SIZEITEMLALR1:= 10000;
       END;
   IF PROC(5)
    THEN
     IF READ(PARMLRO,/,SIZECITEM,SIZEITEM)
      THEN
       BEGIN
        SIZECITEM:= 2000;
```

```
          SIZEITEM := 10000;
         END;
     IF PROC(11)
      THEN
      IF READ(PARMLR1,/,SIZECITEMLR1,SIZEITEMLR1)
        THEN
         BEGIN
          SIZECITEMLR1:= 10000;
          SIZEITEMLR1 := 50000;
         END;
     GERPARSER;
END PROCESSOCOMPLETO;
 IF READ(PARMGRAM,/,NUMNACTERM,NUMTSIMBPROD)
  THEN
   BEGIN
    NUMNACTERM:= 300;
    NUMTSIMBPROD:= 2500;
   END;
 PROCESSCCOMPLETO;
END.
```

A N E X O II

GRAMATICA DE ENTRADA E MENSAGENS DE ERRO

```
ESPACO AUSENTE
E = T ELINHA ;
ELINHA = '+' T ELINHA 1 ;
T = F TLINHA ;
TLINHA = '*' F TLINHA ;
F = '(' E ')' 1 'ID' ;
```

NAOTERMINAIS
REPRESENTACAO INTERNA

NAO TERMINAL	NOME
0	S'
1	E
2	T
3	ELINHA
4	F
5	TLINHA

TERMINAIS
REPRESENTACAO INTERNA

TERMINAL	NOME
6	+
7	*
8	(
9	)
10	ID
11	EOF ($)

PRODUCOES

PRODUCAO	NAO TERMINAL	LISTA DE SIMBOLOS
0	0 ->	1
1	1 ->	2 3
2	2 ->	4 5
3	3 ->	6 2 3
4	3 ->	EPSILON
5	4 ->	8 1 9
6	4 ->	10
7	5 ->	7 4 5

FIRST
TABELA RESULTADO

NAO TERMINAL	LISTA DE TERMINAIS
1	8 10
2	8 10
3	6 EPSILON
4	8 10
5	7

FOLLOW
TABELA RESULTADO

NAO TERMINAL	LISTA DE TERMINAIS
0	11
1	9 11
2	6 9 11
3	9 11
4	7
5	6 9 11

GOTO (LR0)
TABELA RESULTADO

CONJUNTO DE ITEM	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL
0	1	11	2	21	4	31							
2	3	61											
3	5	81											
4	1	101	2	21	4	31							
7	2	111	4	31									
9	4	121											
11	3	141											
12	5	151											

GOTO (LR1)
TABELA RESULTADO

CONJUNTO DE ITEM	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL
0	1	11	2	21	4	31							
2	3	61											
3	5	81											
4	1	101	2	111	4	121							
7	2	131	4	31									
9	4	141											
11	3	161											
12	5	181											
13	3	201											
14	5	211											
17	2	221	4	121									
19	4	231											
22	3	241											
23	5	251											

CONJUNTOS DE ITENS LRO
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POSICAO	PRODUCAO	POSICAO	PRODUCAO	POSICAO	PRODUCAO	POSICAO	PRODUCAO	POSICAO	PRODUCAO	POSICAO	PRODUCAO	POSICAO	PRODUCAO	POSICAO	PRODUCAO	POSICAO	PRODUCAO
0	0	01	1	01	2	01	5	01	6	01									
1	0	11																	
2	1	11	3	01	4	01													
3	2	11	7	01															
4	5	11	1	01	2	01	5	01	6	01									
5	6	11																	
6	1	21																	
7	3	11	2	01	5	01	6	01											
8	2	21																	
9	7	11	5	01	6	01													
10	5	21																	
11	3	21	3	01	4	01													
12	7	21	7	01															
13	5	31																	
14	3	31																	
15	7	31																	

CONJUNTOS DE ITENS LALR1
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POS PONTO	LOOK AHEAD	PRODUCAO	POS PONTO	LOOK AHEAD	PRODUCAO	POS PONTO	LOOK AHEAD	PRODUCAO	POS PONTO	LOOK AHEAD	PRODUCAO	POS PONTO	LOOK AHEAD	PRODUCAO	POS PONTO	LOOK AHEAD
0	0	0	11	1	0	11	2	0	6	2	0	11	5	0	7	6	0	7
1	0	1	11															
2	1	1	9	1	1	11	3	0	9	4	0	9	3	0	11	4	0	11
3	2	1	6	2	1	9	2	1	11	7	0	6	7	0	9	7	0	11
4	5	1	7	1	0	9	2	0	6	2	0	9	5	0	7	6	0	7
5	6	1	7															
6	1	2	9	1	2	11												
7	3 6	1 0	9 7	3	1	11	2	0	6	2	0	9	2	0	11	5	0	7
8	2	2	6	2	2	9	2	2	11									
9	7	1	6	7	1	9	7	1	11	5	0	7	6	0	7			
10	5	2	7															
11	3	2	9	3	2	11	3	0	9	4	0	9	3	0	11	4	0	11
12	7	2	6	7	2	9	7	2	11	7	0	6	7	0	9	7	0	11
13	5	3	7															
14	3	3	9	3	3	11												
15	7	3	6	7	3	9	7	3	11									

CONJUNTOS DE ITENS LR1
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD
0	0	0	11	1	0	11	2	0	6	2	0	11	5	0	7	6	0	7
1	0	1	11															
2	1	1	11	3	0	11	4	0	11									
3	2	1	6	2	1	11	7	0	6	7	0	11						
4	5	1	7	1	0	9	2	0	6	2	0	9	5	0	7	6	0	7
5	6	1	7															
6	1	2	11															
7	3	1	11	2	0	6	2	0	11	5	0	7	6	0	7			
8	2	2	6	2	2	11												
9	7	1	6	7	1	11	5	0	7	6	0	7						
10	5	2	7															
11	1	1	9	3	0	9	4	0	9									
12	2	1	6	2	1	9	7	0	6	7	0	9						
13	3	2	11	3	0	11	4	0	11									
14	7	2	6	7	2	11	7	0	6	7	0	11						
15	5	3	7															
16	1	2	9															
17	3	1	9	2	0	6	2	0	9	5	0	7	6	0	7			
18	2	2	6	2	2	9												
19	7	1	6	7	1	9	5	0	7	6	0	7						
20	3	3	11															
21	7	3	6	7	3	11												
22	3	2	9	3	0	9	4	0	9									
23	7	2	6	7	2	9	7	0	6	7	0	9						
24	3	3	9															

GERADOR PARSER SLR
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACA
0	8	SHIFT	4	10	SHIFT	5								
1	11	REDUCE	0											
2	6	SHIFT	7	9	REDUCE	4	11	REDUCE	4					
3	7	SHIFT	9											
4	8	SHIFT	4	10	SHIFT	5								
5	7	REDUCE	6											
6	9	REDUCE	1	11	REDUCE	1								
7	8	SHIFT	4	10	SHIFT	5								
8	6	REDUCE	2	9	REDUCE	2	11	REDUCE	2					
9	8	SHIFT	4	10	SHIFT	5								
10	9	SHIFT	13											
11	6	SHIFT	7	9	REDUCE	4	11	REDUCE	4					
12	7	SHIFT	9											
13	7	REDUCE	5											
14	9	REDUCE	3	11	REDUCE	3								
15	6	REDUCE	7	9	REDUCE	7	11	REDUCE	7					

GERADOR PARSER LALR1
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO
0	8	SHIFT	4	10	SHIFT	5								
1	11	REDUCE	0											
2	6	SHIFT	7	9	REDUCE	4	11	REDUCE	4					
3	7	SHIFT	9											
4	8	SHIFT	4	10	SHIFT	5								
5	7	REDUCE	6											
6	9	REDUCE	1	11	REDUCE	1								
7	8	SHIFT	4	10	SHIFT	5								
8	6	REDUCE	2	9	REDUCE	2	11	REDUCE	2					
9	8	SHIFT	4	10	SHIFT	5								
10	9	SHIFT	13											
11	6	SHIFT	7	9	REDUCE	4	11	REDUCE	4					
12	7	SHIFT	9											
13	7	REDUCE	5											
14	9	REDUCE	3	11	REDUCE	3								
15	6	REDUCE	7	9	REDUCE	7	11	REDUCE	7					

GERADOR PARSER LR1
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACA
0	8	SHIFT	4	10	SHIFT	5								
1	11	REDUCE	0											
2	6	SHIFT	7	11	REDUCE	4								
3	7	SHIFT	9											
4	8	SHIFT	4	10	SHIFT	5								
5	7	REDUCE	6											
6	11	REDUCE	1											
7	8	SHIFT	4	10	SHIFT	5								
8	6	REDUCE	2	11	REDUCE	2								
9	8	SHIFT	4	10	SHIFT	5								
10	9	SHIFT	15											
11	6	SHIFT	17	9	REDUCE	4								
12	7	SHIFT	19											
13	6	SHIFT	7	11	REDUCE	4								
14	7	SHIFT	9											
15	7	REDUCE	5											
16	9	REDUCE	1											
17	8	SHIFT	4	10	SHIFT	5								
18	6	REDUCE	2	9	REDUCE	2								
19	8	SHIFT	4	10	SHIFT	5								
20	11	REDUCE	3											
21	6	REDUCE	7	11	REDUCE	7								
22	6	SHIFT	17	9	REDUCE	4								
23	7	SHIFT	19											
24	9	REDUCE	3											
25	6	REDUCE	7	9	REDUCE	7								

GRAMATICA DE ENTRADA E MENSAGENS DE ERRO

```
 S = L '=' R ;
 S = R ;
L = '*' R | 'ID' ;
R = L ;
```

NAOTERMINAIS
REPRESENTACAO INTERNA

NAO TERMINAL	NOME
0	S'
1	S
2	L
3	R

TERMINAIS
REPRESENTACAO INTERNA

TERMINAL	NOME
4	=
5	*
6	ID
7	EOF ($)

PRODUCOES

PRODUCAO	NAO TERMINAL		LISTA DE SIMBOLOS		
0	0	->	1		
1	1	->	2	4	3
2	1	->	3		
3	2	->	5	3	
4	2	->	6		
5	3	->	2		

FIRST
TABELA RESULTADO

NAO TERMINAL	LISTA DE TERMINAIS	
1	5	6
2	5	6
3	5	6

FOLLOW
TABELA RESULTADO

NAO TERMINAL	LISTA DE TERMINAIS	
0	7	
1	7	
2	4	7
3	4	7

GOTO (LR0)
TABELA RESULTADO

CONJUNTO DE ITEM	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL
0	1	11	2	21	3	31							
4	2	81	3	71									
6	2	81	3	91									

CONJUNTOS DE ITENS LR0
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POSIPONITO	PRODUCAO	POSIPONITO	PRODUCAO	POSIPONITO	PRODUCAO	POSIPONITO	PRODUCAO	POSIPONITO	PRODUCAO	POSIPONITO	PRODUCAO	POSIPONITO	PRODUCAO	POSIPONITO	PRODUCAO	POSIPONITO	PRODUCAO	POSIPONITO	PRODUCAO	POSIPONITO
0	0	0	1	0	2	0	3	0	4	0	5	0										
1	0	1																				
2	1	1	5	1																		
3	2	1																				
4	3	1	5	0	3	0	4	0														
5	4	1																				
6	1	2	5	0	3	0	4	0														
7	3	2																				
8	5	1																				
9	1	3																				

CONJUNTOS DE ITENS LALR1
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD
0	0 3	0 0	71 71	1 4	0 0	71 71	2	0	71	3	0	41	4	0	41	5	0	71						
1	0	1	71																					
2	1	1	71	5	1	71																		
3	2	1	71																					
4	3 3	1 0	41 71	3 4	1 0	71 71	5	0	41	5	0	71	3	0	41	4	0	41						
5	4	1	41	4	1	71																		
6	1	2	71	5	0	71	3	0	71	4	0	71												
7	3	2	41	3	2	71																		
8	5	1	41	5	1	71																		
9	1	3	71																					

GOTO (LR1)
TABELA RESULTADO

CONJUNTO DE ITEM	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJ ITE FIN
0	1	11	2	21	3	31								
4	2	81	3	71										
6	2	101	3	91										
11	2	101	3	131										

CONJUNTOS DE ITENS LR1
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD
0	0 3	0 0	71 71	1 4	0 0	71 71	2	0	71	3	0	41	4	0	41	5	0	71
1	0	1	71															
2	1	1	71	5	1	71												
3	2	1	71															
4	3 3	1 0	41 71	3 4	1 0	71 71	5	0	41	5	0	71	3	0	41	4	0	41
5	4	1	41	4	1	71												
6	1	2	71	5	0	71	3	0	71	4	0	71						
7	3	2	41	3	2	71												
8	5	1	41	5	1	71												
9	1	3	71															
10	5	1	71															
11	3	1	71	5	0	71	3	0	71	4	0	71						
12	4	1	71															
13	3	2	71															

GERADOR PARSER SLR
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	PRODUCAO	TERMINAL	ACAO	PRODUCAO	TERMINAL	ACAO	PRODUCAO	TERMINAL	ACAO	PRODUCAO	TERMINAL	ACA
0	5	SHIFT	4	6	SHIFT	5								
1	7	REDUCE	0											
2	4	SHIFT	6	7	REDUCE	5								
3	7	REDUCE	2											
4	5	SHIFT	4	6	SHIFT	5								
5	4	REDUCE	4	7	REDUCE	4								
6	5	SHIFT	4	6	SHIFT	5								
7	4	REDUCE	3	7	REDUCE	3								
8	4	REDUCE	5	7	REDUCE	5								
9	7	REDUCE	1											

GERADOR PARSER LALR1
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	PRODUCAO	TERMINAL	ACAO	PRODUCAO	TERMINAL	ACAO	PRODUCAO	TERMINAL	ACAO	PRODUCAO	TERMINAL	ACA
0	5	SHIFT	4	6	SHIFT	5								
1	7	REDUCE	0											
2	4	SHIFT	6	7	REDUCE	5								
3	7	REDUCE	2											
4	5	SHIFT	4	6	SHIFT	5								
5	4	REDUCE	4	7	REDUCE	4								
6	5	SHIFT	4	6	SHIFT	5								
7	4	REDUCE	3	7	REDUCE	3								
8	4	REDUCE	5	7	REDUCE	5								
9	7	REDUCE	1											

GERADOR PARSER LR1
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACA
0	5	SHIFT	4	6	SHIFT	5								
1	7	REDUCE	0											
2	4	SHIFT	6	7	REDUCE	5								
3	7	REDUCE	2											
4	5	SHIFT	4	6	SHIFT	5								
5	4	REDUCE	4	7	REDUCE	4								
6	5	SHIFT	11	6	SHIFT	12								
7	4	REDUCE	3	7	REDUCE	3								
8	4	REDUCE	5	7	REDUCE	5								
9	7	REDUCE	1											
10	7	REDUCE	5											
11	5	SHIFT	11	6	SHIFT	12								
12	7	REDUCE	4											
13	7	REDUCE	3											

GERADOR PARSER SLR
CONFLITOS

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	ACAO	ESTADO PRODUCAO
2	4	REDUCE	5	SHIFT	6

GRAMATICA DE ENTRADA E MENSAGENS DE ERRO

```
ESPACO AUSENTE
E = E '+' E | E '*' E | '(' E ')' | 'ID' ;
```

NAOTERMINAIS
REPRESENTACAO INTERNA

NAO TERMINAL	NOME
0	S'
1	E

TERMINAIS
REPRESENTACAO INTERNA

TERMINAL	NOME
2	+
3	*
4	(
5	)
6	ID
7	EOF ($)

PRODUCOES

PRODUCAO	NAO TERMINAL	LISTA DE SIMBOLOS
0	0 ->	1
1	1 ->	1 2 1
2	1 ->	1 3 1
3	1 ->	4 1 5
4	1 ->	6

FIRST
TABELA RESULTADO

NAO TERMINAL	LISTA DE TERMINAIS
1	4 6

FOLLOW
TABELA RESULTADO

NAO TERMINAL	LISTA DE TERMINAIS
0	7
1	2 3 5 7

GOTO (LRO)
TABELA RESULTADO

CONJUNTO DE ITEM	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL
0	1	1											
2	1	6											
4	1	7											
5	1	8											

CONJUNTOS DE ITENS LRO
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POSIPONTO	PRODUCAO	POSIPONTO	PRODUCAO	POSIPONTO	PRODUCAO	POSIPONTO	PRODUCAO	POSIPONTO	PRODUCAO	POSIPONTO	PRODUCAO	POSIPONTO	PRODUCAO	POSIPONTO	PRODUCAO	POSIPONTO	PRODUCAO
0	0	0	1	0	2	0	3	0	4	0									
1	0	1	1	1	2	1													
2	3	1	1	0	2	0	3	0	4	0									
3	4	1																	
4	1	2	1	0	2	0	3	0	4	0									
5	2	2	1	0	2	0	3	0	4	0									
6	1	1	2	1	3	2													
7	1	1	1	3	2	1													
8	1	1	2	1	2	3													
9	3	3																	

CONJUNTOS DE ITENS LALR1
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD
0	0	0	$	1	0	$	2	0	$	3	0	$	4	0	$	1	0	$
	2	0	)	3	0	)	4	0	)	1	0	)	2	0	)	3	0	)
	4	0	)															
1	0	1	$	1	1	)	1	1	)	1	1	$	2	1	)	2	1	$
	2	1	$															
2	3	1	)	3	1	)	3	1	)	3	1	)	1	0	)	2	0	)
	3	0	)	4	0	)	1	0	)	2	0	)	3	0	)	4	0	)
	1	0	)	2	0	)	3	0	)	4	0	)						
3	4	1	)	4	1	)	4	1	)	4	1	)						
4	1	2	)	1	2	)	1	2	)	1	2	)	1	0	)	2	0	)
	3	0	)	4	0	)	1	0	)	2	0	)	3	0	)	4	0	)
	1	0	)	2	0	)	3	0	)	4	0	)	1	0	)	2	0	)
	3	0	)	4	0	)												
5	2	2	)	2	2	)	2	2	)	2	2	)	1	0	)	2	0	)
	3	0	)	4	0	)	1	0	)	2	0	)	3	0	)	4	0	)
	1	0	)	2	0	)	3	0	)	4	0	)	1	0	)	2	0	)
	3	0	)	4	0	)												
6	1	1	)	1	1	)	1	1	)	2	1	)	2	1	)	2	1	)
	3	2	)	3	2	)	3	2	)	3	2	)						
7	1	1	)	1	1	)	1	1	)	1	1	)	1	3	)	1	3	)
	1	3	)	1	3	)	2	1	)	2	1	)	2	1	)	2	1	)
8	1	1	)	1	1	)	1	1	)	1	1	)	2	1	)	2	1	)
	2	1	)	2	1	)	2	3	)	2	3	)	2	3	)	2	3	)
9	3	3	)	3	3	)	3	3	)	3	3	)						

CONJUNTOS DE ITENS LR1
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POSICAO PONTO	LOOKAHEAD	PRODUCAO	POSICAO PONTO	LOOKAHEAD	PRODUCAO	POSICAO PONTO	LOOKAHEAD	PRODUCAO	POSICAO PONTO	LOOKAHEAD	PRODUCAO	POSICAO PONTO	LOOKAHEAD	PRODUCAO	POSICAO PONTO	LOOKAHEAD
0	0	0	$	1	0	$	2	0	$	3	0	$	4	0	$	1	0	$
	2	0	1	3	0	1	4	0	1	1	0	1	2	0	1	3	0	1
	4	0	3															
1	0	1	$	1	1	1	1	1	1	1	1	$	2	1	1	2	1	1
	2	1	3															
2	3	1	2	3	1	2	3	1	2	1	0	1	2	0	1	3	0	1
	4	0	1	1	0	1	2	0	1	3	0	1	4	0	1	1	0	1
	2	0	3	3	0	3	4	0	3									
3	4	1	$	4	1	3	4	1	$									
			1															
4	1	2	$	1	2	$	1	2	$	1	0	1	2	0	1	3	0	1
	4	0	1	1	0	1	2	0	1	3	0	1	4	0	1	1	0	1
	2	0	3	3	0	3	4	0	3									
5	4	2	$	2	2	$	2	2	$	1	0	1	2	0	1	3	0	1
	4	0	1	4	0	1	4	0	1	3	0	1	4	0	1	1	0	1
	2	0	3	3	0	3	4	0	3									
6	1	1	$	1	1	$	1	1	$	2	1	1	2	1	1	2	1	1
	3	2	1	3	2	1	3	2	1									
7	3	1	$	3	1	$	3	1	$	1	0	1	2	0	1	3	0	1
	4	0	1	4	0	1	4	0	1	3	0	1	4	0	1	1	0	1
	2	0	3	3	0	3	4	0	3									
8	4	1	$	4	1	$	4	1	$									
			1															
9	1	1	$	1	1	$	1	1	$	1	3	1	1	3	1	1	3	1
	2	1	1	2	1	1	2	1	1									
10	1	1	$	1	1	$	1	1	$	2	1	1	2	1	1	2	1	1
	3	1	3	2	1	3	2	1	3									
11	1	2	$	1	2	$	1	2	$	1	0	1	2	0	1	3	0	1
	4	0	1	4	0	1	4	0	1	3	0	1	4	0	1	1	0	1
	2	0	3	3	0	3	4	0	3									
12	2	2	$	2	2	$	2	2	$	1	0	1	2	0	1	3	0	1
	4	0	1	4	0	1	4	0	1	3	0	1	4	0	1	1	0	1
	2	0	3	3	0	3	4	0	3									
13	3	3	$	3	3	$	3	3	$									
			1															
14	1	1	$	1	1	$	1	1	$	2	1	1	2	1	1	2	1	1
	3	2	1	3	2	1	3	2	1									
15	1	1	$	1	1	$	1	1	$	1	3	1	1	3	1	1	3	1
	2	1	1	2	1	1	2	1	1									
16	1	1	$	1	1	$	1	1	$	2	1	1	2	1	1	2	1	1
	2	1	3	2	1	3	2	1	3									
17	3	3	$	3	3	$	3	3	$									
			1															

GERADOR PARSER SLR
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACA
0	4	SHIFT	2	6	SHIFT	3								
1	7	REDUCE	0	2	SHIFT	4	3	SHIFT	5					
2	4	SHIFT	2	6	SHIFT	3								
3	2	REDUCE	4	3	REDUCE	4	5	REDUCE	4	7	REDUCE	4		
4	4	SHIFT	2	6	SHIFT	3								
5	4	SHIFT	2	6	SHIFT	3								
6	2	SHIFT	4	3	SHIFT	5	5	SHIFT	9					
7	2	SHIFT	4	3	REDUCE	1	5	REDUCE	1	7	REDUCE	1		
8	2	SHIFT	4	3	SHIFT	5	5	REDUCE	2	7	REDUCE	2		
9	2	REDUCE	3	3	REDUCE	3	5	REDUCE	3	7	REDUCE	3		

GERADOR PARSER LALR1
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACA
0	4	SHIFT	2	6	SHIFT	3								
1	2	SHIFT	4	3	SHIFT	5	7	REDUCE	0					
2	4	SHIFT	2	6	SHIFT	3								
3	2	REDUCE	4	3	REDUCE	4	5	REDUCE	4	7	REDUCE	4		
4	4	SHIFT	2	6	SHIFT	3								
5	4	SHIFT	2	6	SHIFT	3								
6	2	SHIFT	4	3	SHIFT	5	5	SHIFT	9					
7	2	SHIFT	4	3	SHIFT	5	5	REDUCE	1	7	REDUCE	1		
8	2	SHIFT	4	3	SHIFT	5	5	REDUCE	2	7	REDUCE	2		
9	2	REDUCE	3	3	REDUCE	3	5	REDUCE	3	7	REDUCE	3		

GERADOR PARSER LR1
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACA
0	4	SHIFT	2	6	SHIFT	3								
1	2	SHIFT	4	3	SHIFT	5	7	REDUCE	0					
2	4	SHIFT	7	6	SHIFT	8								
3	2	REDUCE	4	3	REDUCE	4	7	REDUCE	4					
4	4	SHIFT	2	6	SHIFT	3								
5	4	SHIFT	2	6	SHIFT	3								
6	2	SHIFT	11	3	SHIFT	12	5	SHIFT	13					
7	4	SHIFT	7	6	SHIFT	8								
8	2	REDUCE	4	3	REDUCE	4	5	REDUCE	4					
9	2	SHIFT	4	3	SHIFT	5	7	REDUCE	1					
10	2	SHIFT	4	3	SHIFT	5	7	REDUCE	2					
11	4	SHIFT	7	6	SHIFT	8								
12	4	SHIFT	7	6	SHIFT	8								
13	2	REDUCE	3	3	REDUCE	3	7	REDUCE	3					
14	2	SHIFT	11	3	SHIFT	12	5	SHIFT	17					
15	2	SHIFT	11	3	SHIFT	12	5	REDUCE	1					
16	2	SHIFT	11	3	SHIFT	12	5	REDUCE	2					
17	2	REDUCE	3	3	REDUCE	3	5	REDUCE	3					

GOTO (LR1)
TABELA RESULTADO

CONJUNTO DE ITEM	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL
0	1	11											
2	1	6											
4	1	9											
5	1	10											
7	1	14											
11	1	15											
12	1	16											

GERADOR PARSER SLR
CONFLITOS

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	ACAO	ESTADO PRODUCAO
7	2	REDUCE	1	SHIFT	4
7	3	SHIFT	5	REDUCE	1
8	2	REDUCE	2	SHIFT	4
8	3	REDUCE	2	SHIFT	5

GERADOR PARSER LALR
CONFLITOS

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	ACAO	ESTADO PRODUCAO
7	2	REDUCE	1	SHIFT	4
7	3	REDUCE	1	SHIFT	5
8	2	REDUCE	2	SHIFT	4
8	3	REDUCE	2	SHIFT	5

GERADOR PARSER LR1
CONFLITOS

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	ACAO	ESTADO PRODUCAO
9	2	REDUCE	1	SHIFT	4
9	3	REDUCE	1	SHIFT	5
10	2	REDUCE	2	SHIFT	4
10	3	REDUCE	2	SHIFT	5
15	2	REDUCE	1	SHIFT	11
15	3	REDUCE	1	SHIFT	12
16	2	REDUCE	2	SHIFT	11
16	3	REDUCE	2	SHIFT	12

GRAMATICA DE ENTRADA E MENSAGENS DE ERRO

```
S = 'A' A 'C' | 'B' B 'D' | 'A' B 'E' | 'B' A 'E' ;
A = 'C' ;
B = 'C' ;
```

NAOTERMINAIS
REPRESENTACAO INTERNA

NAO TERMINAL	NOME
0	S'
1	S
2	A
3	B

TERMINAIS
REPRESENTACAO INTERNA

TERMINAL	NOME
4	A
5	D
6	B
7	E
8	C
9	EOF ($)

PRODUCOES

PRODUCAO	NAO TERMINAL	LISTA DE SIMBOLOS
0	0 ->	1
1	1 ->	4 2 5
2	1 ->	6 3 5
3	1 ->	4 3 7
4	1 ->	6 2 7
5	2 ->	8
6	3 ->	8

FIRST
TABELA RESULTADO

NAO TERMINAL	LISTA DE TERMINAIS
1	4 6
2	8
3	8

FOLLOW
TABELA RESULTADO

NAO TERMINAL	LISTA DE TERMINAIS
0	9
1	9
2	5 7
3	5 7

GOTO (LR0)
TABELA RESULTADO

CONJUNTO DE ITEM	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL
0	1	1											
2	2	4	3	5									
3	2	8	3	7									

CONJUNTOS DE ITENS LR0
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POSICAO PONTO	PRODUCAO	POSICAO PONTO	PRODUCAO	POSICAO PONTO	PRODUCAO	POSICAO PONTO	PRODUCAO	POSICAO PONTO	PRODUCAO	POSICAO PONTO	PRODUCAO	POSICAO PONTO	PRODUCAO	POSICAO PONTO	PRODUCAO	POSICAO PONTO	PRODUCAO
0	0	0	1	0	2	0	3	0	4	0									
1	0	1																	
2	1	1	3	1	5	0	6	0											
3	2	1	4	1	6	0	5	0											
4	1	2																	
5	3	2																	
6	5	1	6	1															
7	2	2																	
8	4	2																	
9	1	3																	
10	3	3																	
11	2	3																	
12	4	3																	

CONJUNTOS DE ITENS LALR1
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	LOOKAHEAD	PRODUCAO	POS PONTO	A
0	0	0	$	1	0	$	2	0	$	3	0	$	4	0	$			
1	0	1	$															
2	1	1	$	3	1	$	5	0	)	6	0	)						
3	2	1	$	4	1	$	6	0	)	5	0	)						
4	1	2	$															
5	3	2	$															
6	5	1	)	5	1	)	6	1	)	6	1	)						
7	2	2	$															
8	4	2	$															
9	1	3	$															
10	3	3	$															
11	2	3	$															
12	4	3	$															

GOTO (LR1)
TABELA RESULTADO

CONJUNTO DE ITEM	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL
0	1	11													
2	2	41	3	51											
3	2	81	3	71											

CONJUNTOS DE ITENS LR1
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POS PONTO	LOCK AHEAD	PRODUCAO	POS PONTO	LOCK AHEAD	PRODUCAO	POS PONTO	LOCK AHEAD	PRODUCAO	POS PONTO	LOCK AHEAD	PRODUCAO	POS PONTO	LOCK AHEAD	PRODUCAO	POS PONTO	A
0	0	0	91	1	0	91	2	0	91	3	0	91	4	0	91			
1	0	1	91															
2	1	1	91	3	1	91	5	0	51	6	0	71						
3	2	1	91	4	1	91	6	0	51	5	0	71						
4	1	2	91															
5	3	2	91															
6	5	1	51	6	1	71												
7	2	2	91															
8	4	2	91															
9	5	1	71	6	1	51												
10	1	3	91															
11	3	3	91															
12	2	3	91															
13	4	3	91															

GERADOR PARSER SLR
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO
0	4	SHIFT	2	6	SHIFT	3								
1	9	REDUCE	0											
2	8	SHIFT	6											
3	8	SHIFT	6											
4	5	SHIFT	9											
5	7	SHIFT	10											
6	5	REDUCE	5	7	REDUCE	5								
7	5	SHIFT	11											
8	7	SHIFT	12											
9	9	REDUCE	1											
10	9	REDUCE	3											
11	9	REDUCE	2											
12	9	REDUCE	4											

GERADOR PARSER LALR1
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACA
0	4	SHIFT	2	6	SHIFT	3								
1	9	REDUCE	0											
2	8	SHIFT	6											
3	8	SHIFT	6											
4	5	SHIFT	9											
5	7	SHIFT	10											
6	5	REDUCE	5	7	REDUCE	5								
7	5	SHIFT	11											
8	7	SHIFT	12											
9	9	REDUCE	1											
10	9	REDUCE	3											
11	9	REDUCE	2											
12	9	REDUCE	4											

GERADOR PARSER LR1
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACA
0	4	SHIFT	2	6	SHIFT	3								
1	9	REDUCE	0											
2	8	SHIFT	6											
3	8	SHIFT	9											
4	5	SHIFT	10											
5	7	SHIFT	11											
6	5	REDUCE	5	7	REDUCE	6								
7	5	SHIFT	12											
8	7	SHIFT	13											
9	7	REDUCE	5	5	REDUCE	6								
10	9	REDUCE	1											
11	9	REDUCE	3											
12	9	REDUCE	2											
13	9	REDUCE	4											

A N E X O III

GERADOR PARSER SLR
CONFLITOS

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	ACAO	ESTADO PRODUCAO
6	5	REDUCE	6	REDUCE	5
6	7	REDUCE	6	REDUCE	5

GERADOR PARSER LALR
CONFLITOS

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	ACAO	ESTADO PRODUCAO
6	5	REDUCE	6	REDUCE	5
6	7	REDUCE	6	REDUCE	5

GRAMATICA DE ENTRADA E MENSAGENS DE ERRO

```
ESPACO_AUSENTE
CONTEXT_FREE_GRAMMAR
    = 'SPACES'          # A NULL GRAMMAR
    | CONTEXT_FREE_GRAMMAR RULE 'SPACES'
    ;
RULE
    = LEFT_PART '=' ALTERNATIVES ';'
    ;
ALTERNATIVES
    = RIGHT_PART
    | ALTERNATIVES '|' RIGHT_PART
    ;
LEFT_PART
    = NONTERMINAL 'SPACES'
    ;
RIGHT_PART
    = 'SPACES'          # AN EMPTY RIGHT PART
    | RIGHT_PART TERMINAL 'SPACES'
    | RIGHT_PART NONTERMINAL 'SPACES'
    ;
NONTERMINAL = 'IDENTIFIER' ;
TERMINAL = 'STRING' ;
```


NAOTERMINAIS
REPRESENTACAO INTERNA

NAO TERMINAL	NONE
0	S'
1	CONTEXT_FREE_GRAMMAR
2	RULE
3	LEFT_PART
4	ALTERNATIVES
5	RIGHT_PART
6	NONTERMINAL
7	TERMINAL

TERMINAIS
REPRESENTACAO INTERNA

TERMINAL	NOME
8	SPACES
9	=
10	;
11	|
12	IDENTIFIER
13	STRING
14	EOF ($)

PRODUCOES

PRODUCAO	NAC TERMINAL		LISTA DE SIMBOLOS			
0	0	->	1			
1	1	->	8			
2	1	->	1	2	8	
3	2	->	3	9	4	10
4	3	->	6	8		
5	4	->	5			
6	4	->	4	11	5	
7	5	->	8			
8	5	->	3	7	8	
9	5	->	5	6	8	
10	6	->	12			
11	7	->	13			

FIRST

TABELA RESULTADO

NAO TERMINAL	LISTA DE TERMINAIS
1	8
2	12
3	12
4	8
5	8
6	12
7	13

FOLLOW

TABELA RESULTADO

NAO TERMINAL	LISTA DE TERMINAIS
0	14
1	12 14
2	8
3	9
4	10 11
5	10 11 12 13
6	8
7	8

GOTO (LR0)
TABELA RESULTADO

CONJUNTO DE ITEM	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL
0	1	11											
1	2	31	3	41	6	51							
8	4	101	5	111									
11	6	161	7	151									
14	5	181											
18	6	161	7	151									

GOTO (LR1)
TABELA RESULTADO

CONJUNTO DE ITEM	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL	CONJUNTO ITEM FINAL	NAO TERMINAL
0	1	11											
1	2	31	3	41	6	51							
8	4	101	5	111									
11	6	161	7	151									
14	5	181											
18	6	161	7	151									

CONJUNTOS DE ITENS LRO
TABELA RESULTADO

CONJUNTO DE ITENS	PRODUCAO	POSIPONTOI	PRODUCAO	POSIPONTOI	PRODUCAO	POSIPONTOI	PRODUCAO	POSIPONTOI	PRODUCAO	POSIPONTOI	PRODUCAO	POSIPONTOI	PRODUCAO	POSIPONTOI	PRODUCAO	POSIPONTOI	PRODUCAO	POSIPONTOI	PRODUCAO
0	0	01	1	01	2	01													
1	0	11	2	11	3	01	4	01	10	01									
2	1	11																	
3	2	21																	
4	3	11																	
5	4	11																	
6	10	11																	
7	2	31																	
8	3	21	5	01	6	01	7	01	8	01	9	01							
9	4	21																	
10	3	31	6	11															
11	5	11	8	11	9	11	11	01	10	01									
12	7	11																	
13	3	41																	
14	6	21	7	01	8	01	9	01											
15	8	21																	
16	9	21																	
17	11	11																	
18	6	31	8	11	9	11	11	01	10	01									
19	8	31																	
20	9	31																	

CONJUNTO DE ITENS	PRODUCAO	POS FON TO	LOOKAHEAD	PRODUCAO	POS FON TO	LOOKAHEAD	PRODUCAO	POS FON TO	LOOKAHEAD	PRODUCAO	POS FON TO	LOOKAHEAD	PRODUCAO	POS FON TO	LOOKAHEAD	PRODUCAO	POS FON TO	LOOKAHEAD
0	0	0	141	1	0	141	2	0	141	1	0	121	2	0	121			
1	0	1	141	2	1	121	2	1	141	3	0	81	4	0	91	10	0	81
2	1	1	121	1	1	141												
3	2	2	121	2	2	141												
4	3	1	81															
5	4	1	91															
6	10	1	81															
7	2	3	121	2	3	141												
8	3 5 3	2 0 0	81 111 121	5 6 9	0 0 0	101 111 121	6 7 7	0 0 0	101 131 111	7 8 6	0 0 0	101 131 111	6 9 9	0 0 0	101 131 111	9 7	0 0	101 121
9	4	2	91															
10	3	3	81	6	1	101	6	1	111									
11	5 9	1 1	101 101	5 9	1 1	111 111	6 9	1 1	101 121	6 9	1 1	111 131	6 11	1 0	121 81	6 10	1 0	131 81
12	7	1	101	7	1	111	7	1	121	7	1	131						
13	3	4	81															
14	6 8 6	2 0 0	101 111 121	6 9 9	2 0 0	111 111 121	7 7	0 0	101 131	6 8	0 0	101 131	9 9	0 0	101 131	7 7	0 0	111 121
15	8	2	101	6	2	111	8	2	121	6	2	131						
16	9	2	101	9	2	111	9	2	121	9	2	131						
17	11	1	81															
18	6 9	3 1	101 101	6 9	3 1	111 111	6 9	1 1	101 121	6 9	1 1	111 131	6 11	1 0	121 81	6 10	1 0	131 81
19	8	3	101	6	3	111	6	3	121	6	3	131						
20	9	3	101	9	3	111	9	3	121	9	3	131						

CONJUNTOS DE ITENS LR1
TABELA RESULTADO

CONJUNTO DE ITENS I	PRODUCAO	POS PONTO	LOOKI AHEADI	PRODUCAO	POS PONTO	LOOKI AHEADI	PRODUCAO	POS PONTO	LOOKI AHEADI	PRODUCAO	POS PONTO	LOOKI AHEADI	PRODUCAO	POS PONTO	LOOKI AHEADI	PRODUCAO	POS PONTO	LOOKI AHEADI
0	0	0	141	1	0	141	2	0	141	1	0	121	2	0	121			
1	0	1	141	2	1	121	2	1	141	3	0	81	4	0	91	10	0	81
2	1	1	121	1	1	141												
3	2	2	121	2	2	141												
4	3	1	81															
5	4	1	91															
6	10	1	81															
7	2	3	121	2	3	141												
8	3 5 8	2 0 0	81 111 121	5 6 9	0 0 0	101 111 121	6 7 7	0 0 0	101 131 111	7 8 8	0 0 0	101 131 111	8 9	0 0	101 131 111	9 7	0 0	101 121
9	4	2	91															
10	3	3	81	6	1	101	6	1	111									
11	5 9	1 1	101 101	5 9	1 1	111 111	6 9	1 1	101 121	8 9	1 1	111 131	8 11	1 0	121 81	8 10	1 0	131 81
12	7	1	101	7	1	111	7	1	121	7	1	131						
13	3	4	81															
14	6 8 8	2 0 0	101 111 121	6 9 9	2 0 0	111 111 121	7 7	0 0	101 131	8 8	0 0	101 131	9	0	101 131	7 7	0 0	111 121
15	8	2	101	8	2	111	8	2	121	8	2	131						
16	9	2	101	9	2	111	9	2	121	9	2	131						
17	11	1	81															
18	6 9	3 1	101 101	6 9	3 1	111 111	8 9	1 1	101 121	8 8	1 1	111 131	8 11	1 0	121 81	8 10	1 0	131 81
19	8	3	101	8	3	111	8	3	121	8	3	131						
20	9	3	101	9	3	111	9	3	121	9	3	131						

GERADOR PARSER SLR
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACA
0	8	SHIFT	2											
1	14	REDUCE	0	12	SHIFT	6								
2	12	REDUCE	1	14	REDUCE	1								
3	8	SHIFT	7											
4	9	SHIFT	8											
5	8	SHIFT	9											
6	8	REDUCE	10											
7	12	REDUCE	2	14	REDUCE	2								
8	8	SHIFT	12											
9	9	REDUCE	4											
10	10	SHIFT	13	11	SHIFT	14								
11	10	REDUCE	5	11	REDUCE	5	13	SHIFT	17	12	SHIFT	6		
12	10	REDUCE	7	11	REDUCE	7	12	REDUCE	7	13	REDUCE	7		
13	8	REDUCE	3											
14	8	SHIFT	12											
15	8	SHIFT	19											
16	8	SHIFT	20											
17	8	REDUCE	11											
18	10	REDUCE	6	11	REDUCE	6	13	SHIFT	17	12	SHIFT	6		
19	10	REDUCE	8	11	REDUCE	8	12	REDUCE	8	13	REDUCE	8		
20	10	REDUCE	9	11	REDUCE	9	12	REDUCE	9	13	REDUCE	9		

GERADOR PARSER LALR1
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACA
0	8	SHIFT	2											
1	12	SHIFT	6	14	REDUCE	0								
2	12	REDUCE	1	14	REDUCE	1								
3	8	SHIFT	7											
4	9	SHIFT	3											
5	8	SHIFT	9											
6	8	REDUCE	10											
7	12	REDUCE	2	14	REDUCE	2								
8	8	SHIFT	12											
9	9	REDUCE	4											
10	10	SHIFT	13	11	SHIFT	14								
11	12	SHIFT	6	13	SHIFT	17	10	REDUCE	5	11	REDUCE	5		
12	10	REDUCE	7	11	REDUCE	7	12	REDUCE	7	13	REDUCE	7		
13	8	REDUCE	3											
14	8	SHIFT	12											
15	8	SHIFT	19											
16	8	SHIFT	20											
17	8	REDUCE	11											
18	12	SHIFT	6	13	SHIFT	17	10	REDUCE	6	11	REDUCE	6		
19	10	REDUCE	8	11	REDUCE	8	12	REDUCE	8	13	REDUCE	8		
20	10	REDUCE	9	11	REDUCE	9	12	REDUCE	9	13	REDUCE	9		

GERADOR PARSER LR1
TABELA RESULTADO

ESTADO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACAO	ESTADO PRODUCAO	TERMINAL	ACA
0	8	SHIFT	2											
1	12	SHIFT	6	14	REDUCE	0								
2	12	REDUCE	1	14	REDUCE	1								
3	8	SHIFT	7											
4	9	SHIFT	8											
5	8	SHIFT	9											
6	8	REDUCE	10											
7	12	REDUCE	2	14	REDUCE	2								
8	8	SHIFT	12											
9	9	REDUCE	4											
10	10	SHIFT	13	11	SHIFT	14								
11	12	SHIFT	6	13	SHIFT	17	10	REDUCE	5	11	REDUCE	5		
12	10	REDUCE	7	11	REDUCE	7	12	REDUCE	7	13	REDUCE	7		
13	8	REDUCE	3											
14	8	SHIFT	12											
15	8	SHIFT	19											
16	8	SHIFT	20											
17	8	REDUCE	11											
18	12	SHIFT	6	13	SHIFT	17	10	REDUCE	6	11	REDUCE	6		
19	10	REDUCE	8	11	REDUCE	8	12	REDUCE	8	13	REDUCE	8		
20	10	REDUCE	9	11	REDUCE	9	12	REDUCE	9	13	REDUCE	9		

A N E X O IV

GRAMATICA DE ENTRADA E MENSAGENS DE ERRO

```
S = C C ;
C = 'C' C | 'D' ;
```

(IV.1)

GRAMATICA DE ENTRADA E MENSAGENS DE ERRO

```
ESPACO AUSENTE
E = T ELINHA ;
ELINHA = '+' T ELINHA | ;
T = F TLINHA ;
TLINHA = '*' F TLINHA ;
F = '(' E ')' | 'ID' ;
```

(IV.2)

GRAMATICA DE ENTRADA E MENSAGENS DE ERRO

```
ESPACO AUSENTE
CONTEXT_FREE_GRAMMAR
    = 'SPACES'          # A NULL GRAMMAR
    | CONTEXT_FREE_GRAMMAR RULE 'SPACES'
    ;
RULE
    = LEFT_PART '=' ALTERNATIVES ';'
    ;
ALTERNATIVES
    = RIGHT_PART
    | ALTERNATIVES '|' RIGHT_PART
    ;
LEFT_PART
    = NONTERMINAL 'SPACES'
    ;
RIGHT_PART
    = 'SPACES'          # AN EMPTY RIGHT PART
    | RIGHT_PART TERMINAL 'SPACES'
    | RIGHT_PART NONTERMINAL 'SPACES'
    ;
NONTERMINAL = 'IDENTIFIER' ;
TERMINAL = 'STRING' ;
```

(IV.3)

GRAMATICA DE ENTRADA E MENSAGENS DE ERRO

```
P  =  B  ;
B  =  'B'  L  'E'  ;
L  =  L  ';'  C
   |  LD  ';'  C
   |  C  ;
LD  =  LD  ';'  D
   |  D  ;
D  =  D  ','  V
   |  'INT'  V  ;
C  =  B
   |  V  ':='  E
   |  'IF'  E  'TH'  C  ;
E  =  A  '='  A
   |  A  ;
A  =  A  '+'  T
   |  T  ;
T  =  T  '*'  F
   |  F  ;
F  =  '('  E  ')'
   |  V
   |  'IF'  E  'TH'  F  'EL'  F  ;
V  =  'A'  ;
```

(IV.4)

GRAMATICA DE ENTRADA E MENSAGENS DE ERRO

```
PROG = BLOCO ;
BLOCO = 'BEGIN' LISTA 'END' ;
LISTA = LISTA ';' COM
      | LISTA_DECL ';' COM
      | COM ;
LISTA_DECL = LISTA_DECL ';' DECL
           | DECL ;
DECL = 'INTEGER' VAR
     | DECL ',' VAR ;
COM = BLOCO
    | AT
    | 'GOTO' ID
    | ROTULO '.' COM
    | 'IF' EXPR 'THEN' COM
    | 'RESULT' EXPR ;
VAR = ID
    | ID '<' LISTA_EXPR '>' ;
ROTULO = ID ;
LISTA_EXPR = LISTA_EXPR ';' EXPR
           | EXPR ;
EXPR = EXPR_ARIT '=' EXPR_ARIT
     | EXPR_ARIT ;
EXPR_ARIT = EXPR_ARIT '+' TERMO
          | TERMO ;
TERMO = TERMO '*' FATOR
      | FATOR ;
FATOR = '(' EXPR ')'
      | PRIMARIO ;
PRIMARIO = BLOCO
         | CONST
         | VAR ;
CONST = 'K' ;
ID = 'IO' ;
AT = VAR ',' AT ',' EXPR
   | VAR '.=' EXPR ;
```

(IV.5)

GRAMATICA DE ENTRADA E MENSAGENS DE ERRO

```
 PROG = PROG_HEAD BLOCK '.' ;
PROG_HEAD = 'PROGRAM' 'ID' '(' SEQ_FILE_ID ')' ';' ;
SEQ_FILE_ID = SEQ_FILE_ID ',' FILE_ID | FILE_ID ;
FILE_ID = 'ID' ;
BLOCK = LABEL_PART CONST_PART TYPE_PART VAR_PART PROC_FUNC_PART STAT_PART ;
LABEL_PART =  | 'LABEL' SEQ_LABEL ';' ;
SEQ_LABEL = SEQ_LABEL ',' LABEL | LABEL ;
LABEL = UNS_INTG ;
UNS_INTG = 'INTEGER' ;
CONST_PART =  | 'CONST' SEQ_CONST_DEF ';' ;
SEQ_CONST_DEF = SEQ_CONST_DEF ';' CONST_DEF | CONST_DEF ;
CONST_DEF = 'ID' '=' CONST ;
CONST = UNS_INTG | SIGN UNS_INTG | CONST_ID | SIGN CONST_ID ;
SIGN = '+' | '-' ;
CONST_ID = 'ID' ;
TYPE_PART =  | 'TYPE' SEQ_TYPE_DEF ';' ;
SEQ_TYPE_DEF = SEQ_TYPE_DEF ';' TYPE_DEF | TYPE_DEF ;
TYPE_DEF = 'ID' '=' TYPE ;
TYPE = SIMPLE_TYPE | STRUCT_TYPE ;
SIMPLE_TYPE = SCALAR_TYPE | SUBRANGE_TYPE | TYPE_ID ;
SCALAR_TYPE = '(' SEQ_ID ')' ;
SEQ_ID = SEQ_ID ',' 'ID' | 'ID' ;
SUBRANGE_TYPE = CONST '..' CONST ;
TYPE_ID = 'ID' ;
STRUCT_TYPE = ARRAY_TYPE | SET_TYPE ;
ARRAY_TYPE = 'ARRAY' '[' SEQ_INDEX_TYPE ']' 'OF' COMPON_TYPE ;
SEQ_INDEX_TYPE = SEQ_INDEX_TYPE ',' INDEX_TYPE | INDEX_TYPE ;
INDEX_TYPE = SIMPLE_TYPE ;
COMPON_TYPE = TYPE ;
CASE_LABEL = CONST ;
SET_TYPE = 'SET' 'OF' BASE_TYPE ;
BASE_TYPE = SIMPLE_TYPE ;
VAR_PART =  | 'VAR' SEQ_VAR_DEC ';' ;
SEQ_VAR_DEC = SEQ_VAR_DEC ';' VAR_DEC | VAR_DEC ;
VAR_DEC = SEQ_ID ':' TYPE ;
PROC_FUNC_PART =  | SEQ_PROC_FUNC_DEC ;
SEQ_PROC_FUNC_DEC = PROC_FUNC_DEC ';' SEQ_PROC_FUNC_DEC | PROC_FUNC_DEC ';' ;
PROC_FUNC_DEC = PROC_DEC | FUNC_DEC ;
PROC_DEC = PROC_HEAD BLOCK ;
```

(IV.6)

```
PROC_HEAD = 'PROCEDURE' 'ID' ';' ! 'PROCEDURE' 'ID' '(' SEQ_FPAR_SEC ')' ';' ;
SEQ_FPAR_SEC = SEQ_FPAR_SEC ';' FPAR_SEC ! FPAR_SEC ;
FPAR_SEC = PAR_GROUP ! 'VAR' PAR_GROUP ! 'FUNCTION' PAR_GROUP !
           'PROCEDURE' SEQ_ID ;
PAR_GROUP = SEQ_ID ':' TYPE_ID ;
FUNC_DEC = FUNC_HEAD BLOCK ;
FUNC_HEAD = 'FUNCTION' 'ID' ':' RESULT_TYPE ';' !
            'FUNCTION' 'ID' '(' SEQ_FPAR_SEC ')' ':' RESULT_TYPE ';' ;
RESULT_TYPE = TYPE_ID ;
STAT_PART = COMPOUND_STAT ;
STATEMENT = UNLAB_STAT ! LABEL ':' UNLAB_STAT ;
UNLAB_STAT = SIMPLE_STAT ! STRUCT_STAT ! PROCED_STAT ;
SIMPLE_STAT = ASSIGN_STAT ! EMPTY_STAT ! GOTO_STAT ;
ASSIGN_STAT = VARIABLE ':=' EXPRESSION ! FUNCTION_ID ':=' EXPRESSION ;
VARIABLE = ENTIRE_VARIABLE ! COMPON_VARIABLE ;
ENTIRE_VARIABLE = VARIABLE_ID ;
VARIABLE_ID = 'ID' ;
COMPON_VARIABLE = INDEXED_VARIABLE ;
INDEXED_VARIABLE = ARRAY_VARIABLE '[' SEQ_EXPRES ']' ;
SEQ_EXPRES = SEQ_EXPRES ',' EXPRESSION ! EXPRESSION ;
ARRAY_VARIABLE = VARIABLE ;
EXPRESSION = SIMPLE_EXPRES ! SIMPLE_EXPRES RELOP SIMPLE_EXPRES ;
RELOP = '=' ! '<>' ! '<' ! '<=' ! '>=' ! '>' ! 'IN' ;
SIMPLE_EXPRES = TERM ! SIGN TERM ! SIMPLE_EXPRES ADDING_OP TERM ;
ADDING_OP = '+' ! '-' ! 'OR' ;
TERM = FACTOR ! TERM MULT_OP FACTOR ;
MULT_OP = '*' ! 'DIV' ! 'MOD' ! 'AND' ;
FACTOR = VARIABLE ! UNS_CONST ! '(' EXPRESSION ')' ! FUNC_DESIGN ! SET !
         'NOT' FACTOR ;
UNS_CONST = UNS_INTG ! CONST_ID ;
FUNC_DESIGN = FUNCTION_ID ! FUNCTION_ID '(' SEQ_ACTPAR ')' ;
SEQ_ACTPAR = SEQ_ACTPAR ',' ACTPAR ! ACTPAR ;
FUNCTION_ID = 'ID' ;
SET = '[' ELEMENT_LIST ']' ;
ELEMENT_LIST = SEQ_ELEMENT ! EMPTY ;
SEQ_ELEMENT = SEQ_ELEMENT ',' ELEMENT ! ELEMENT ;
ELEMENT = EXPRESSION ! EXPRESSION '..' EXPRESSION ;
PROCED_STAT = PROCED_ID ! PROCED_ID '(' SEQ_ACTPAR ')' ;
PROCED_ID = 'ID' ;
ACTPAR = EXPRESSION ! VARIABLE ;
GOTO_STAT = 'GOTO' LABEL ;
EMPTY_STAT = EMPTY ;
EMPTY = ;
STRUCT_STAT = COMPOUND_STAT ! CONDITION_STAT ! REPET_STAT ;
```

```
COMPOUND_STAT = 'BEGIN' SEQ_STAT 'END' ;
SEQ_STAT = SEQ_STAT ';' STATEMENT | STATEMENT ;
CONDITION_STAT = IF_STAT | CASE_STAT ;
IF_STAT = 'IF' EXPRESSION 'THEN' STATEMENT |
          'IF' EXPRESSION 'THEN' STATEMENT 'ELSE' STATEMENT ;
CASE_STAT = 'CASE' EXPRESSION 'OF' SEQ_CASE_LIST_ELM 'END' ;
SEQ_CASE_LIST_ELM = SEQ_CASE_LIST_ELM ';' CASE_LIST_ELM | CASE_LIST_ELM ;
CASE_LIST_ELM = CASE_LABEL_LIST ':' STATEMENT | EMPTY ;
CASE_LABEL_LIST = SEQ_CASE_LABEL ;
SEQ_CASE_LABEL = SEQ_CASE_LABEL ',' CASE_LABEL | CASE_LABEL ;
REPET_STAT = WHILE_STAT | REPEAT_STAT | FOR_STAT ;
WHILE_STAT = 'WHILE' EXPRESSION 'DO' STATEMENT ;
REPEAT_STAT = 'REPEAT' SEQ_STAT 'UNTIL' EXPRESSION ;
FOR_STAT   = 'FOR' CONTROL_VAR ':=' FOR_LIST 'DO' STATEMENT ;
FOR_LIST = INIT_VAL 'TO' FINAL_VAL | INIT_VAL 'DOWNTO' FINAL_VAL ;
INIT_VAL = EXPRESSION ;
FINAL_VAL = EXPRESSION ;
CONTROL_VAR = VARIABLE_ID ;
```

BIBLIOGRAFIA

1. AHO, A.V. & ULLMAN, J.D.- Principles of compiler design 2.ed. Reading, Mass., Addison- Wesley, 1977

2. HORNING, J.J.- "LR grammars and analysers" In : BAUER,F.L. & EICKEL, J.- Compiler construction: an advanced course 2.ed. New York, N.Y., Springer-Verlag, 1974, p.85-108.

3. GRIES,D.- Compiler construction for digital computers. New York, N.Y., John Wiley & Sons Inc., 1971

4. AHO, A.V. & ULLMAN,J.D.- The theory of parsing,translation and compiling, Volume 1: Parsing. Englewood Cliffs,N.J., Prentice-Hall, 1972

5. AHO, A.V. & ULLMAN, J.D.- The theory of parsing, translation and compiling, Volume 2 : Compiling. Englewood,Cliffs , N.J., Prentice Hall, 1973

6. KNUTH, D.E.- On the translation of languages from left to right. Information and Control, Volume 8,607-639, 1965

7. ANDERSON, T.; EVE, J.; HORNING, J.J.- Efficient LR(1) parsers. Acta Informatica, Volume, 12-39, 1973.

8. DE HAMMER, F.L.- Simple LR(k) grammars. Communications of the ACM , Volume 14, 453-460, 1971

9. JOLIAT, M.L.: On the reduced matrix representation of the LR(k) parser tables. Computer Systems Group Techn.Rep., University of Toronto, Rep. CSRG-28, 1973.

10. PAGER, D.- A solution to an open problem by Knuth. Information and Control, Volume 17, 462-473, 1970

11. JACKSON, M.A.- Principles of program design. 2.ed. New York, N.Y., Academic Press, 1975.

12. DE HAMMER, F.L.- "Review of formalisms and notations" In: BAUER, F.L. & EICKEL, J.- Compiler construction : an advanced course. 2 ed. New York, N.Y., Springer-Verlag, 1974, p.50-53.

13. BURROUGHS CORPORATION- Algol language reference manual. 1974.