UNUVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE MATEMÁTICA ESTATÍSTICA E CIÊNCIA DA COMPUTAÇÃO

LL - LINGUAGEM PARA LINGUISTAS

JOSÉ ARMANDO VALENTE

ORIENTADOR
Prof. Dr. Nelson Castro Machado

Dissertação apresentada ao INSTITUTO DE MATEMÁTICA ESTATÍSTICA E CIÊNCIA DA COMPUTAÇÃO para obtenção do titulo de "MESTRE".

CAMPINAS
DEZEMBRO DE 1973

SINOPSE

Este trabalho descreve a estrutura e implementação do sistema LL (LINGUAGEM PARA LINGUISTA), que possibilita ao usuário a mecanização dos testes de modelos gramaticais, utilizando as facilidades de um computador de grande porte em "Time - Sharing".

O sistema caracteriza-se pela sua generalidade aceitando C-gramáticas sem qualquer restrição, e opera de modo conversacional, sendo de fácil manuseio mesmo para pesquisadores não acostumados ao uso de computador, devido à adoção de notação e terminologia familiares ao linguista, bem como da apresentação dos resultados em forma de árvore de derivação.

Aos meus pais

e avos

AGRADECIMENTOS

Os meus agradecimentos ao Professor JACQUES COHEN pela proposição, apoio e insentivo, fatores marcantes para a realização deste trabalho.

Ao Professor NELSON MACHADO pela orientação, leitura e revisão do texto.

Ao Professor CARLOS FRANCHI e LIGIA LEDERMAN pelas sugestoes e conhecimentos transmitidos no tocante à linguística.

Aos meus amigos e colegas do Instituto e em especial acs colegas de Departamento TADAO, FERRARI e VANINI, pelas críticas, oportunas sugestoes e valiosa colaboraçao que dispensaram a este trabalho

INDICE

CAPÍTULO 1

INTRODUÇÃO

A Linguística moderna visa construir uma teoria da estrutura da linguagem humana, cabendo pois ao linguista :

a) - encontrar um conjunto de regras (gramática) e aliar a esse conjunto propriedades gerais que permitam utilizá-lo como base para o estudo da estrutura de qualquer lingua humana.

b) - dada uma determinada lingua, encontrar uma gramática que gere "todas e somente" /5/ as frases da lingua, isto é , uma gramática que permita descrever qualquer expressão válida dentro da lingua, mas que não admita expressoes que de alguma forma venham fugir à sua estrutura.

c) - montar a gramática de maneira a permitir simular os processos utilizados no aprendizado e emprego da linguagem pelo homem.

Uma vez proposta uma gramática dentro da teoria linguística moderna para descrever uma linguagem, o sucesso fica sujeito à verificação de que realmente "todas e somente" as frases dessa linguagem são geradas por essa gramática.

Compete então ao linguista lançar mão de um método de análise exaustiva por inspeção da gramática, a fim de determinar por experimentação o que esta sendo gerado. Este procedimento é evidentemente cansativo, tedioso e, se feito sem o mínimo auxílio mecânico, muito condicionado à experiência e perseverança do próprio pesquisador.

Considerando que este processo é possível de ser formalizado e descrito por um número finito de açoes elementa-

res, o presente trabalho, utiliza os recursos teóricos da Ciência da Computação e as facilidades de equipamento atualmente disponíveis /10/, para estabelecer um sistema utilizável por linguistas, a fim de mecanizar o teste de modelos gramaticais, permitindo-lhe concentrar-se nos aspéctos mais relevantes de sua atividade.

O sistema recebeu a denominação de LL - LINGUAGEM PARA LINGUISTAS, sendo implementado em FORTRAN IV para o DEC - 10 /9/, utilizando um GERADOR DE COMPILADORES /7/ que muito facilitou a implantação.

O sistema LL, basicamente, esta dividido em - tres programas principais :

a) - Definição da gramática - recebe os dados que definem uma gramática : conjunto dos símbolos não terminais, conjunto dos símbolos terminais, axioma (símbolo especial) e o conjunto de regras.

b) - Analisador - analisa sintaticamente uma frase dada de acordo com a gramática previamente definida.

c) - Saída dos resultados - informa através de uma árvore de derivação ou de uma mensagem se a frase é respectivamente - gerada ou não pela gramática.

As informaçoes a serem processadas por cada um desses programas não fornecidos através de comandos que constituem a linguagem LL, e com os quais o linguista pode se comunicar com o sistema.

Como uma breve descrição do que se segue temos: no capítulo 2 são introduzidas algumas noçoes fundamentais a respeito da teoria empregada no desenvolvimento deste trabalho; no capítulo 3 é dada a descrição da linguagem LL em geral, bem como o formato dos comandos da LL ; o capítulo 4 apresenta alguns exemplos de utilização do sistema LL, aplicando-o a problemas propostos; no ca-

pítulo 5 são fornecidos os detalhes de implementação e finalmente no capítulo 6 são feitas algumas considerações a respeito de desenvolvimento futuros e conclusões.

Para o leitor interessado apenas na utilização do LL, recomendamos a leitura dos capítulos 3 e 4 que constituem um manual de uso do sistema.

CAPÍTULO 2

GRAMÁTICA — NOÇÕES BÁSICAS

Este capítulo introduz ao leitor os conceitos teóricos, notação e terminologia a partir dos quais este trabalho foi desenvolvido. Dado o seu caráter de referência, devemos salientar que não se trata de uma exposição minuciosa, mas sim reunião de resultados que virão a ser posteriormente utilizados.

2.1 - LINGUÍSTICA - CONSIDERAÇÕES GERAIS

Os estudos da linguagem tem merecido sempre , desde a antiguidade a atenção de pesquisadores. Até meados do século XX porém, a linguagem foi estudada não como uma disciplina autônoma mas sempre associada e comprometida a escolas filosóficas , a crítica literárias e outras atividades.

A linguística moderna, graças a Chomsky,vem de se corporificar como uma disciplina independente, dotada de caráter e objetivos científicos, buscando formar uma teoria que permita abordar todos os tipos (existentes ou não) de manifestações linguísticas, dentro de uma única metodologia isenta de escolismos ou elitismos culturais. Mais ainda, encarando a linguagem (e a expressão por meio da linguagem) como intrinsecamente inerentes à natureza humana, pretende Chomsky estudá-la não simplesmente como um fim em si mesma, mas como uma atividade onde se podem colher observações importantes sobre mecanismos de percepção, aprendizagem e pensamento /3/.

Em seus trabalhos, Chomsky empresta uma das mais importantes e originais contribuições à teoria da linguagem : depois de salientar o aspécto criador da linguagem, a partir da ob-

servação simples de que o falante é capaz de produzir e compreender noções que jamais ouviu anteriormente, introduz a noção de competência, como sendo o sistema de regras que o falante interiorizou e que lhe permite com recursos finitos, produzir infinitas frases.

Uma gramática se define então como um modelo dessa competência. Cumpre ao linguista elaborar tal modelo : uma gramática explícita , com exigências de rigor e precisão matemáticos na formalização da teoria sintática; uma gramática projetiva no sentido de possibilitar a extensão dos resultados válidos para um conjunto de frases, tomada em teste de adequação empírica, a todas as frases gramaticais da lingua. Chomsky destaca essa tarefa em algumas questões fundamentais /5/.

2.2 - GRAMÁTICA E LINGUAGEM

No item anterior introduzimos a idéia intuitiva do conceito de gramática, sob o ponto de vista linguístico, nosso objetivo, neste item, é a formalização deste conceito com o intuito de torná-lo operacional /6/ , /4/.

2.2.1 - DEFINIÇÃO DE GRAMÁTICA

Um alfabeto ou vocabulário é um conjunto V,não vazio e finito, de elementos. Qualquer arranjo com repetição (concatenação) de elementos de um alfabeto V é chamada de uma sequência ou palavra. A sequência vazia será denotada por $\in$.

O comprimento $|X|$ de uma sequência X é o número de elementos que a compõe. Assim, $|\in| = 0$.

Dado um alfabeto V, representa-se por V^* o conjunto das sequências finitas formadas por elementos de V, e por V^+ a $V^* - \in$.

Definição 1 :- Dado um alfabeto V, uma regra ou produção é uma relação do tipo :

$$x \longrightarrow y$$

onde $x \in V^+$ e $y \in V^*$, x é chamado de parte esquerda e y de parte direita da regra, e a relação de V^+ em V^* é uma relação de substituição.

O processo de aplicação de regras é chamado de derivação.

Definição 2 :- Dado um alfabeto V, define-se gramática como o quadripleto : G (S) = (V_N , V_T , S , P) onde :

V_N - Conjunto dos não terminais , é o conjunto de elementos que pertence a V e aparecem como lado esquerdo de uma regra.

V_T - Conjunto dos terminais , é o conjunto de elementos que pertencem a V e não pertence a V_N.

$V_N \cup V_T = V$ e $V_N \cap V_T = \emptyset$ (vazio)

S - Símbolo inicial ou axioma, é o elemento que pertence a V_N e a partir do qual começam todas as derivações.

P - É um conjunto não vazio e finito de regras.

Definição 3 :- Uma linguagem de uma gramática é o conjunto de sequências formadas por elementos de V_T (V_T^+) - que são derivados de S. A um elemento desse conjunto damos o nome de sentença ou frase.

Definição 4 :- Dada uma gramática G (S), se as regras em P são relações /16/ de V_N em V^*, a gramática é chamada - livre de contexto ou C-gramática (CG)

As C-gramáticas são assim chamadas porque, ao contrário do que acontece com os outros tipos de gramática, o elemento da parte esquerda da regra pode ser substituido pela sequência da parte direita independente do contexto no qual ele aparece.

Definição 5 :- Uma frase gerada por uma gramática é ambigua se existem duas ou mais árvores de derivação /17/- para esta frase. Uma gramática é ambigua se contém fra-- ses ambiguas.

2.2.2 - NOTAÇÃO E TERMINOLOGIA LINGUÍSTICA

A notação comumente utilizada para definição- do conjunto P nas teorias de gramática formal é a notação BNF /13/ ou variante simplificada.

Nos meios linguísticos, todavia usa-se escrever as regras de uma gramática utilizando uma outra notação, introduzida por Chomsky /5/ e desenvolvida em Back /1/.

1) - Dado um alfabeto V, os símbolos + ou $\cap$ ou & indicam a concatenação de elementos de V.

Exemplo 2.1 para o alfabeto $\{a,b,c\}$, as sequências como:

a a b , b c a , etc

são indicados por :

a + a + b ou a$\cap$a$\cap$b , etc

2) - Regras envolvendo sequências como :

A $\longrightarrow$ B C D

B $\longrightarrow$ C D

C $\longrightarrow$ B

são representadas então por :

A $\longrightarrow$ B + C + D

B $\longrightarrow$ C + D

$$C \rightarrow B$$

3) - As regras do tipo :

$$A \rightarrow B + C$$

$$A \rightarrow B$$

que apresentam idêndica parte esquerda, são resumidas, através do uso de $\{\ \}$ como :

$$A \rightarrow \left\{ \begin{matrix} B + C \\ B \end{matrix} \right\}$$

ou na forma linearizada como :

$$A \rightarrow \{B + C,\ B\}$$

4) - Regras como:

$$A \rightarrow B + C$$

$$A \rightarrow B$$

cujas partes direitas diferem pela presença ou ausência de um símbolo ou sequência de símbolos, o uso de parênteses circundando - o(s) símbolo(s) opcional(is) sintetiza ainda mais a representação- das regras :

$$A \rightarrow B\ (C)$$

Exemplo 2.2 - o conjunto das regras :

$$B \rightarrow C$$

$$B \rightarrow C\ D$$

$$B \rightarrow D\ R\ S$$

$$B \rightarrow D\ R$$

$$B \rightarrow D\ R\ D$$

para o vocabulário $V = \{B, C, D, R, S\}$, são representados por :

$$B \rightarrow \left\{ \begin{matrix} C\ (D) \\ D + \left\{ \begin{matrix} R + S \\ R\ (D) \end{matrix} \right\} \end{matrix} \right\}$$

OBS : considerando ser este trabalho voltado principalmente para - linguistas, passaremos a utilizar esta notação no decorrer do tex

to.

2.3 - ANÁLISE SINTÁTICA DE UMA FRASE

Para uma gramática dada, analisar sintaticamente uma frase consiste em verificar se esta frase pertence ou não à gramática, isto é, se existe uma árvore de derivação para a frase dada.

Um programa ou algoritmo para análise ou seja, um analisador, processa ou analisa uma frase percorrendo-a da esquerda para a direita ou vice-versa e buscando, em síntese, a árvore de derivação para esta frase, utilizando as regras da gramática.

Os analisadores são classificados em duas categorias : "Botton-up" e Top-down" /13/. O termo refere-se a maneira como eles constroem a árvore, se desde os elementos terminais até o axioma, ou desde o axioma até os elementos terminais.

2.3.1 - ANALIZADOR "BOTTON-UP"

O analisador "BOTTON-UP" constroi a árvore de derivação partindo da frase e tentando reduzi-la até atingir o axioma. Assim o analisador deve procurar determinar o lado direito ou pseudo lado direito de uma regra e substitui-lo pelo lado esquerdo (reduzir). Todos os analisadores "Botton-up" trabalham segundo este procedimento, diferindo uns dos outros, em dois aspéctos : no número e posiçoes de símbolos usados para determinar o lado direito da regra, e no modo como as regras e tabelas são estruturadas - no próprio reconhecedor.

2.3.2. - ANALISADOR "TOP-DOWN"

O analisador "Top-down" constroi a árvore de - derivação a partir do axioma, buscando obter a frase dada através-

da aplicação, por tentativa, de regras adequadas dentre as que - constituem a gramática. Quando na aplicação de determinada regra - se atinge uma situação que não mais permita continuar a derivação, o método prevê a volta à situação imediatamente anterior ("Back-track"), para tentar aplicar uma outra regra, e assim por diante.

Dentro do esquema geral de analisadores " Top-down ", existem diversas dessemelhança, relacionadas principalmente com os recursos utilisados para se proceder a escolha da regra mais adequada, visando diminuir ou eliminar o "Back-track".

CAPÍTULO 3

O SISTEMA LL

Neste capítulo descrevemos as características gerais e comandos da linguagem LL, bem como as mensagens e as opções de saída fornecidas pelo sistema LL.

3.1 - OBJETIVOS E CARACTERÍSTICAS

O sistema LL é, basicamente um programa, que recebe como entrada um programa escrito na linguagem LL, no qual o usuário define uma gramática e as frases a serem analisadas, e fornece como saída a árvore de derivação relativa à análise da sentença dada ou uma mensagem informando que a sentença não é gerada pela gramática.

Além desta característica bastante desejável e útil aos linguistas, o sistema proporciona ainda, as seguintes facilidades :

a) - operação conversacional:- este aspécto é bastante importante pois, uma vez que o linguista tenha um sistema conversacional ("TIME-SHARING") à sua disposição, uma melhor iteração com a máquina é possível o maior número de testes podem ser efetuados.

b) - operação utilizando arquivos :- O programa escrito pelo usuário em LL, pode ser armazenado em um arquivo (utilizando o dispositivo como disco, fita, etc.) e, para ser executado basta fornecer o nome deste arquivo ao sistema.

c) - notação e terminologia familiares ao linguista : - levando-se em consideração este fato, o usuário dispensa menor tempo para conhecer e manusear o sistema, sentindo-se mais seguro em

em relação a transcrição do seu problema utilizando a linguagem LL.

d) - opções de saída :- para uma frase analisada, o usuário tem duas opções de resposta ; a árvore de derivação ou a listagem das regras ordenadas segundo a ordem com que foram aplicadas.

Como uma breve descrição do sistema, para um programa fornecido, as seguintes etapas são efetuadas para o seu processamento:

a) - para a gramática são construidas tabela do tipo dicionário, onde são armazenados todos os elementos pertencentes aos conjuntos de não terminais e terminais, e as regras são armazenadas utilizando uma estrutura de árvore /18/.

b) - para implementação do analisador, é utilizado o método do "Top-down" com "Back-track" automático (vide item 2.3.2) armazenando as informações para posterior construção da árvore de derivação. Caso se constate que a frase é ambigua (vide item .2.2) são construidas sequencialmente todas as árvores de derivação.

c) - baseado nas informações fornecidas pelo analisador é construida, utilizando-se um algoritmo, a (s) árvore (s) de derivação correspondente.

3.2 - COMANDOS DA LINGUAGEM LL

3.2.1 - NORMAS GERAIS

a) - um programa escrito na linguagem LL, deve começar e terminar com o símbolo $ (dollar)

b) - todos os comandos devem terminar com o símbolo ; (ponto e virgula)

c) - os comandos podem ser escritos em qualquer posição (coluna 1 até 72).

3.2.2 - DEFINIÇÃO DA GRAMÁTICA

A gramática é definida por uma quádrupla de comandos : VN, VT, AXIOMA e REGRAS, que devem ser codificadas nesta - ordem obrigatoriamente.

3.2.2.1 - COMANDO-VN

Função :

Definir os elementos pertencentes ao conjunto de não terminais da gramática.

Forma Geral :

VN : lista de elementos ;

Descrição :

O comando VN consiste do mnemônico VN : seguidos dos elementos que compõemo conjunto de não terminais da gramática, separados por um ou mais brancos. Um elemento é uma sequência de até 50 caractéres alfanuméricos sendo que o primeiro deve ser obrigatoriamente um alfabético.

3.2.2.2 - COMANDO - VT

Função :

Definir os elementos pertencentes ao conjunto de terminais da gramática.

Forma Geral :

VT : lista de elementos ;

Descrição :

O comando VT consiste do mnemônico VT: seguido dos elemen

tos que compõem o conjunto de terminais da gramática separados por um ou mais brancos. Um elemento é uma sequência de até 50 caracte-- res alfanuméricos sendo que o primeiro deve ser obrigatoriamente um alfabético.

3.2.2.3 - COMANDO - AXIOMA

Função :

Definir o elemento especial (axioma) do conjunto VN, a partir do qual são iniciadas todas as derivações.

Forma Geral :

AXIOMA : elemento ;

Descrição :

O comando AXIOMA consiste do mnemônico AXIOMA : seguido - de um elemento pertencente ao conjunto de não terminais que define o axioma da gramática.

3.2.2.4 - COMANDO - REGRAS

Função :

Definir as regras pertencentes ao conjunto de regras da gramática (notação linguística - vide item 2.2.2)

Forma Geral :

```
REGRAS :  regra 1;
          regra 2;
          :
          regra n;
```

Descrição :

O comando REGRAS consiste do mnemônico REGRAS : seguido - das regras que definem o conjunto de regras da gramática. Uma regra deve ser escrita utilizando-se a notação do linguista linearizada - (vide item 2.2.2), a qual deve ser codificada escrevendo-se um elemento pertencente ao conjunto dos não terminais, em seguida os sinais → (menos e maior) e a parte direita da regra. Para a codificação da parte direita devem ser utilizados os sinais : + (mais) para indicar a concatenação de elementos, [ ] (colchetes) para indicar- o ou exclusivo, ( )(parênteses) para indicar a ocorrência de opcional e, (vírgula) para separar as sequências de elementos escritos - entre colchetes. Nota-se que a regra :

P → (A)

é equivalente a :

P → [A, ∈] onde ∈ representa a sequência vazia.

EXEMPLO 3.1 : para a regra escrita segundo a notação do linguista :

$$P \rightarrow \left\{ \begin{matrix} B + C \\ (A) \end{matrix} \right\}$$

deve ser codificada como :

P → [B + C , (A)]

3.2.3 - COMANDO - ANALISE

Função :

Definir o conjunto de frases a serem analisadas segundo a gramática definida através dos comandos do item 3.2.2

Forma Geral :

```
ANALISE : frase 1 ;
          frase 2 ;
          frase n ;
```

Descrição :

O comando ANÁLISE consiste do mnemônico ANALISE: seguido das frases que devem ser analisadas baseado na gramática fornecida pelo linguista. Uma frase é uma sequência de elementos pertencentes ao conjunto de terminais (podendo ser vazia), separados por um ou mais brancos.

Exemplo 3.2 : para as frases : O homem virou o barco, menino, e a frase vazia, o comando de ser codificado como :

```
ANALISE : O HOMEM VIROU O BARCO ;
          MENINO ;
                 ;
```

3.2.4 - OPÇÕES DE SAÍDA

Para cada frase analisada com sucesso, é dado ao usuário a escolha de uma dentre 3 opções de saídas. Antes de imprimir a análise o sistema interroga :

ÁRVORE (Ø) - REGRAS (1) - AMBAS (2)

O usuário para obter a árvore de derivação deverá fornecer o número Ø (zero); para obter a listagem das regras ordenadas segundo a ordem com que foram utilizadas para a análise da frase, deverá fornecer o número 1 e para obter a árvore e a listagem das regras, o número 2 (dois).

3.2.5 - COMANDO - IMPRESSÃO DAS REGRAS

Função :

Imprimir todas as regras definidas através do comando RE-

GRAS, em notação do linguista de modo explicito.

Forma Geral :

IMPRESSÃO DE REGRAS;

Descrição :

O comando para impressão das regras consiste dos mnemônicos IMPRESSÃO DE REGRAS, colocados obrigatoriamente nesta ordem.

3.3 - MENSAGENS DE ERRO

O sistema LL, devido à suas características funcionais, é bastante útil ao usuário quando utilizado em "Time-Sharing" (via terminal). Para melhor interação usuário-máquina, e menor tempo dispendido na interpretação das mensagens de erro, procuramos fornecer o máximo de informação possível a fim de que o usuário possa facilmente identificar o erro e corrigi-lo.

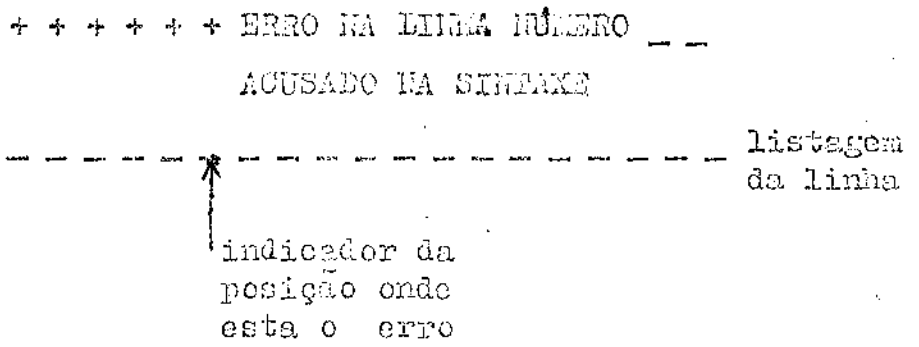
Dois tipos de mensagens são fornecidas pelo sistema LL, sendo ambas fatais (encerram o processamento do programa):

a) - erro de sintaxe - o comando não esta corretamente escrito.

b) - erro de semântica - encontrado na execução de um comando.

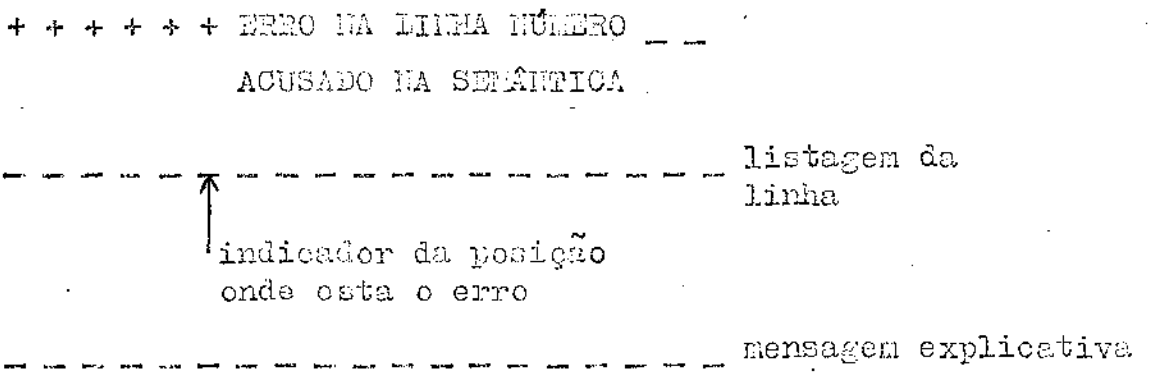
3.3.1 - ERRO DE SINTAXE

Para este tipo de erro, a mensagem fornecida consta da listagem da linha onde foi encontrado o erro e a posição deste na linha.



3.3.2 - ERRO DE SEMÂNTICA

Para esse tipo de erro a mensagem fornecida consta da listagem da linha onde foi encontrado o erro, a posição deste na linha e uma mensagem explicativa do tipo de erro (por exemplo, se o símbolo da parte esquerda de uma regra não pertence a V_N, se ocorreu o transbordamento (overflow) de um vetor, etc).



CAPÍTULO 4

UTILIZAÇÃO DO SISTEMA LL

Neste capítulo descrevemos como utilizar o sistema LL para resolução de problemas que frequentemente surgem em pesquisas linguísticas. Dois exemplos serão desenvolvidos : o primeiro ilustra a utilização do LL de modo conversacional e o segundo através de arquivo.

4.1 - ENTRADA NO SISTEMA LL

O usuário para utilizar o sistema LL deve primeiramente fornecer informaçoes ao computador de modo a ter o sistema LL a sua disposição. Para tanto deve datilografar :

. RUN LL [1]

Como resposta, o sistema LL, agora sob controle, imprime a mensagem:

& & & & & & & & & & & & & & & & & & & & &

IMPRIMA COMVE PARA CONVERSACIONAL

OU O NOME DO ARQUIVO

& & & & & & & & & & & & & & & & & & & & &

*

---

1 - os símbolos ou sentenças grifadas significam informaçoes fornecidas pelo sistema ao usuário.

4.2. - UTILIZAÇÃO

O usuário deve fornecer em seguida, uma das , dentre as tres opções :

a) - palavra CONVE - se desejar utilizar o sistema LL de modo conversacional.

b) - palavra EXIT - se desejar encerrar o processamento (sair do sistema)

c) - um nome de até 5 caracteres alfanumérico sendo que o primeiro deve ser obrigatoriamente um alfabético. Este nome refere-se a um arquivo onde foi armazenado um programa previamente escrito em linguagem LL. Deve ser notado, pelas condições dos itens a e b, que este nome deve ser diferente de CONVE ou EXIT.

4.2.1 - CONVE - CONVERSACIONAL

Se a palavra CONVE foi datilografada, o sistema fornece o símbolo ? (interrogação), indicando que o usuário deve datilografar um comando ou continuação de um comando (quando um comando completo não der para ser escrito numa linha).

O sistema reconhece e analisa esta instrução - somente quando ela estiver completa (isto é, encontrar um ;) passando a executa-la se estiver corretamente escrita, caso contrário fornece uma mensagem de erro e encerra o processamento.

Terminada a execução do referido comando, um novo símbolo ? (interrogação) é fornecido pelo sistema, um novo comando deve ser datilografado e assim por diante.

Quando o usuário desejar encerrar o programa , como resposta ao símbolo ? deve datilografar o símbolo $ (dollar). O sistema encerra este programa e volta a fornecer as mensagens do item 4.1.

4.2.2 - EXIT - ENCERRAR O PROCESSAMENTO

Quando não existir mais programas para ser executado pelo sistema LL, o processamento deve ser encerrado datilografando a palavra EXIT. Se o usuário desejar novemente utilizar o sistema LL, deve datilografar a informação

. RUN LL

4.2.3 - NOME DE UM ARQUIVO

Para utilizar o sistema LL através de arquivo, o usuário deve inicialmente criar um arquivo, utilizando os recursos do computador, no qual deve ser armazenado um ou mais programas escritos em linguagem LL, a ser executado.

Os comando são lidos deste arquivo pelo sistema LL, analisados e executados um a um . Terminada a execução dos programas deste arquivo, o sistema volta a fornecer as mensagens do item 4.1.

4.3 EXEMPLO 1

Como exemplo de utilização, vamos considerar a gramática G1(s) definida por :

$V_N = \{S, NP, AUX, UP, V, DET, N, M\}$

$V_T = \{O, A, UMA, PODE, DEVE, SINCERIDADE, MENINO, BARCO, VIRAR, BUSCAR\}$

AXIOMA = S

REGRAS =
$S \longrightarrow NP + (AUX) + UP$
$VP \longrightarrow V + NP$
$NP \longrightarrow (DET) + N$
$DET \longrightarrow [O, A, UMA]$
$AUX \longrightarrow M$
$M \longrightarrow [PODE, DEVE]$

$$N \longrightarrow [\text{SINCERIDADE, MENINO, BARCO}]$$
$$V \longrightarrow [\text{VIRAR, BUSCAR}]$$

O apêndice D, mostra a utilização do sistema - LL de modo conversaçional. O apêndice E, mostra a utilização do - sistema LL para mesma gramática utilizando o programa armazenado - no arquivo de nome ANAE.

4.4 - EXEMPLO 2

Como outro exemplo, vamos considerar a gramática recursiva à direita G2(S) definida por :

$$V_N = \{S, X\}$$
$$V_T = \{A\}$$
$$\text{AXIOMA} = S$$
$$\text{REGRAS} = \{ S \longrightarrow X$$
$$X \longrightarrow [A + X, A] \}$$

No apêndice F temos a utilização do sistema LL de modo conversacional, analisando frases dessa gramática.

4.5 - EXEMPLO 3

Nesse exemplo utilizamos o sistema LL de modo-conversacional, com intuito de mostrar uma condição de erro fornecida pelo sistema.

Na definição da regra :

$$y \longrightarrow X + X ;$$

o sinal + não foi colocado e o sistema acusou um erro de sintaxe , encerrando o processamento. Esse exemplo encontra-se no apêndice G

CAPÍTULO 5

IMPLEMENTAÇÃO DO SISTEMA LL

Neste capítulo discutimos as principais técnicas, recursos e estruturas de dados utilizados na implementação do sistema LL. Não procuramos apresentar uma descrição detalhada dos algoritmos utilizados mas sim ressaltar o papel de cada ponto na performance geral do sistema, analisando vantagens, desvantagens e eventuais alternativas que poderiam ter sido empregadas.

5.1 - CONSIDERAÇÕES GERAIS

O desempenho final de qualquer sistema de programação fundamenta-se principalmente na estrutura de dados e recursos empregados para sua criação. O sistema LL utiliza dois recursos fundamentais : Gerador de Compiladores /7/ e FORTRAN- Não- Determinístico (ND FORT) /8/, /12/.

O Gerador de Compiladores pelo seu carater bastante automático, contribui sobremaneira para aliviar a programação de determinadas tarefas, além de emprestar uma estrutura adequada - ao sistema LL.

5.1.1 - GERADOR DE COMPILADORES

É um programa que aceita como entrada um conjunto de dados que definem uma linguagem (isto é, os tipos de símbolos aceitos e as combinações permitidas desses símbolos), gerando na - saída um conjunto de programas interligados, os quais constituem um compilador para a linguagem descrita na entrada. O compilador compõem-se basicamente, de quatro programas : (Fig. 5.1).

a) - Analisador léxico - cuja finalidade é verificar se os símbolos utilizados são os permitidos e se estão corretamente escritos, baseados em informaçoes sobre os símbolos básicos.

b) - Analisador sintático - cuja finalidade é verificar - se a sintaxe dos comandos da linguagem estão corretamente formados, baseados na B N F - especificada para a linguagem.

c) - Recuperador de erro - cuja função é analisar erros - cometidos durante a execução e emitir mensagens explicativas apro— priadas.

d) - Rotinas semânticas - que serão executadas à medida - que o analisador reconhece os símbolos de entradas.

Os tres primeiros.programas são construidos automaticamente pelo gerador ; as rotinas semânticas devem ser programadas e incorporadas ao sistema pelo usuário.

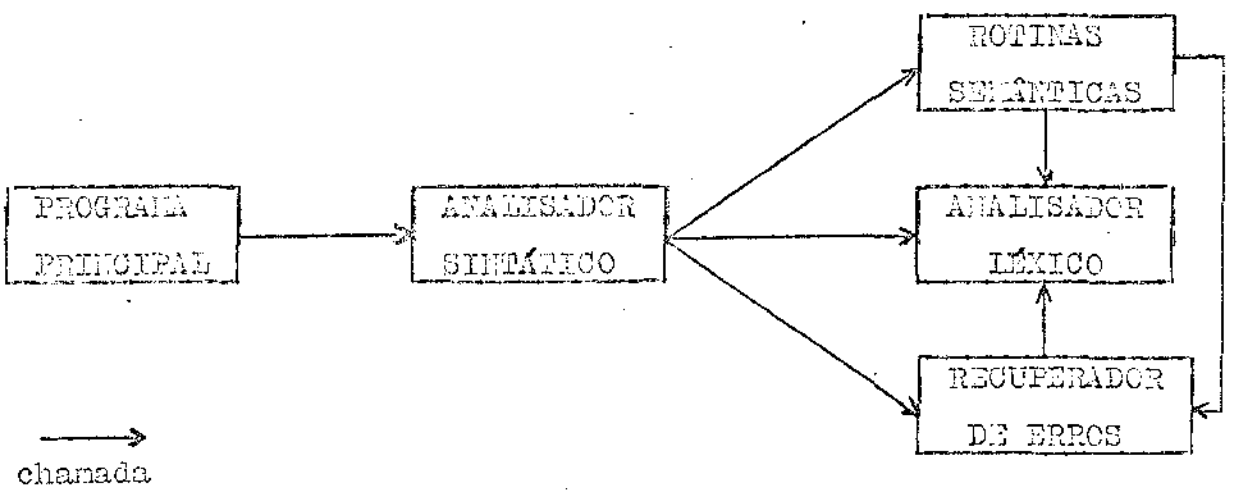


Fig. 5.1

5.1.2 - ESQUEMA DO SISTEMA LL

Um comando da linguagem LL, suprido pelo usuário, deve ser observadas, para o seu processamento, as seguintes - etapas :

a) - Deve-se proceder à verificação de que a sequência de caracteres não apresenta quaisquer símbolos não permitidos na linguagem.

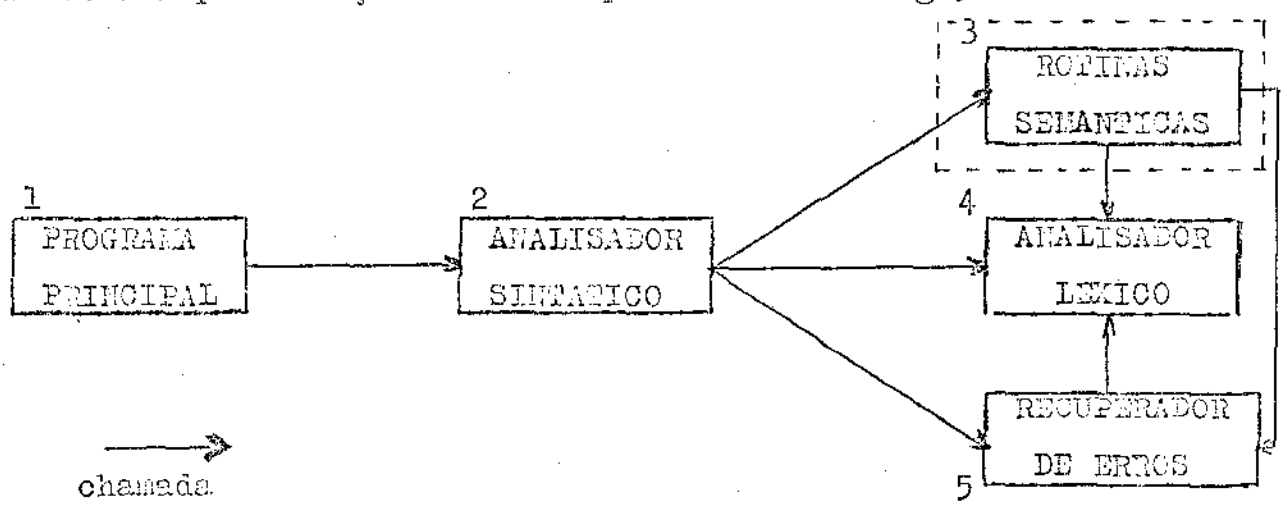
b) - assumindo que a sequência de caracteres de entrada pode pertencer a linguagem LL, deve-se proceder ao reconhecimento da sequência de entrada e a consequente determinação dos que exatamente o usuário espera que o sistema faça.

c) - uma vez determinada a natureza do comando, o sistema deve executar a ação propriamente como resposta à solicitação de entrada.

d) - na hipótese de que por algum motivo não se possa concluir a ação desejada, o sistema deve prover mensagem adequada, comunicando ao usuário o término anormal de processamento.

Estas fases de processamento são naturalmente obtidas utilizando-se o Gerador de Compiladores, acrescentando-se a comodidade de geração automática de rotinas, e da comprovada eficiência do sistema.

O diagrama básico do Gerador de Compiladores aplicado à implementação do LL e apresentado na Fig 5.2 :



1 - gerado automaticamente

2 - " "

3 - programado pelo usuário

Fig. 5.2

4 - gerado automaticamente

5 - " " "

A descrição sintática (BNF) para geração do analisador sintático.(principal programa do sistema) encontra-se no apêndice A.

A construção da rotina semântica (opçoes), que passa agora a ser o nosso principal interesse é, esquematizada na fig 5.3

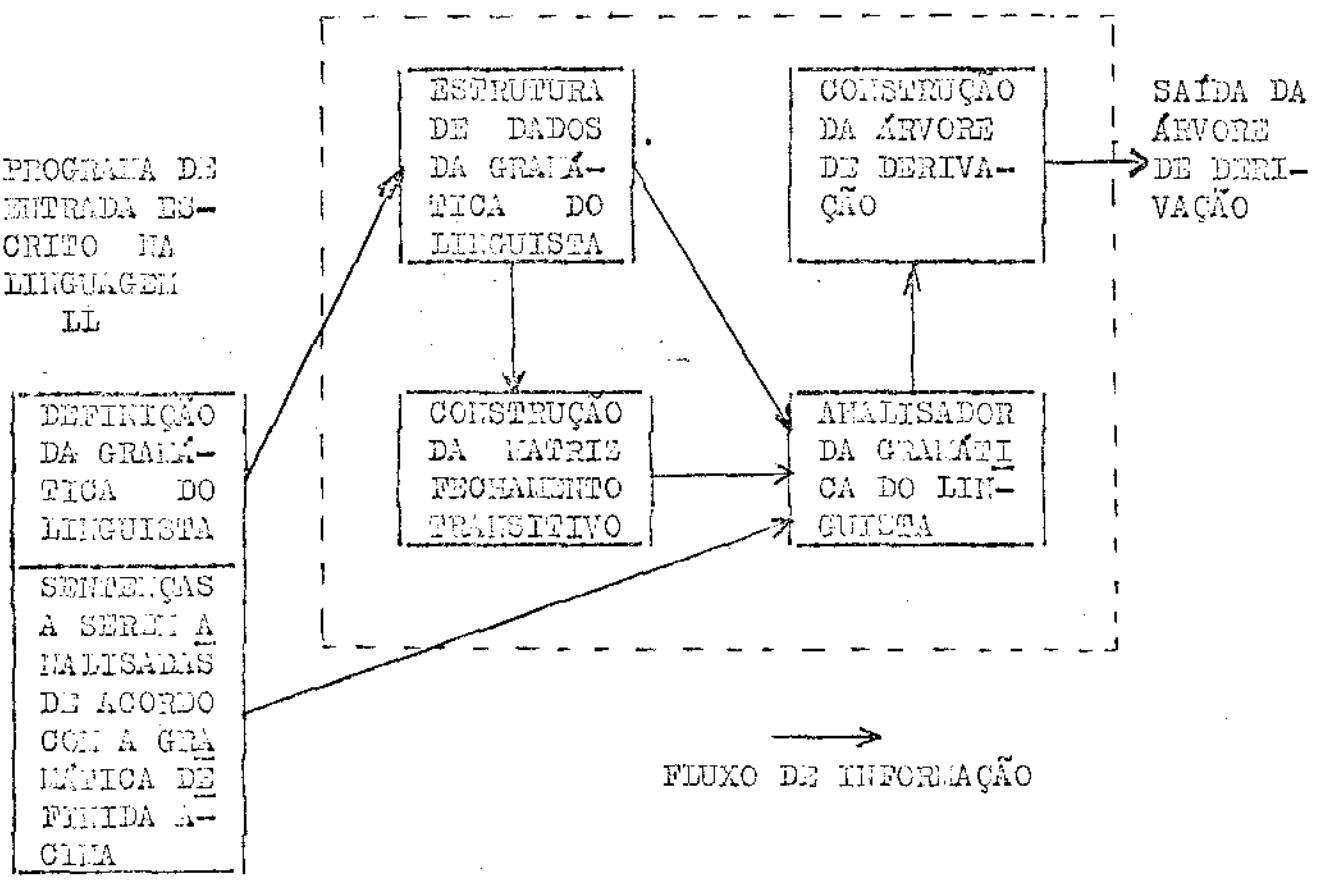


Fig. 5.3

5.2 — ROTINAS SEMÂNTICAS

5.2.1. - ARMAZENAMENTO DA GRAMÁTICA DO LINGUISTA

Como foi visto no item 3.2.2 a gramática é definada através de quatro comandos VN , VT , AXIOMA ou REGRAS, fornecido pelo usuário, nesta ordem, são construidas as seguintes estruturas de dados :

5.2.1.1 - VOCABULARIO

Os comandos VN e VT provocam, cada qual por sua vez, a mesma ação: armazenamento de elementos de gramática.

A cada não terminal ou terminal atribui-se um código (número negativo correspondente à posição ocupada no conjunto). Isto feito, os elementos são armazenados num discionário tipo "HASHING" /18/ e são recuperados consultando-se os endereços correspondentes, armazenados numa tabela auxiliar.

Ocorrências múltiplas de um símbolo em VN ou em VT são automaticamente eliminadas; um elemento que aparece simultaneamente em VN e VT gera uma mensagem de erro, causando o término do processamento.

5.2.1.2 - AXIOMA

O comando AXIOMA armazena em uma variável o código do elemento de VN assim definido. A ausência do elemento de - AXIOMA dentre os elementos de VN é condição de erro.

5.2.1.3 - REGRAS

As regras enumeradas no comando REGRAS, ainda na fase de preparação de dados para o analisador, são armazenados através de uma estrutura de árvore /18/, onde os nós, são os códigos dos elementos. Antes de procedermos a construção desta árvore , são

efetuados dois testes de consistência :

a) - se o elemento da parte esquerda pertence a VT

b) - se os elementos que compõem a parte direita foram definidos anteriormente.

Satisfeitas estas condições, passamos a construção da árvore :

a) - regras ou partes de regras definidas por uma sequência de elementos utilizando a notação + (vide itens 2.2.2 e 3.2.2.4)

b) - regras ou partes de regras definidas pelas notações [ ] e ( ) respectivamente (itens 2.2.2 e 2.2.2.4)

Para o caso a são construídos nós contendo as seguintes informações :

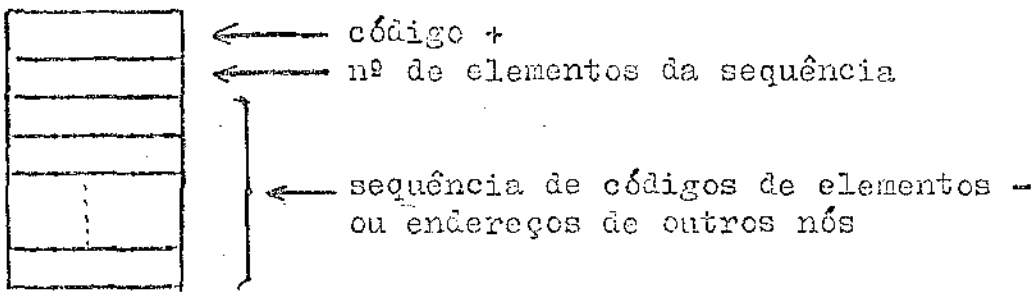


Fig 5.4

No caso b, nós relativos à notação [ ], são do tipo :

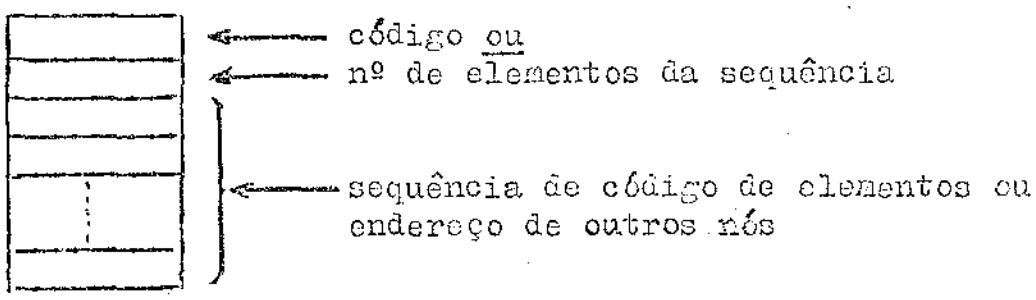


Fig 5.5

E os relativos a ( ) tem a estrutura :

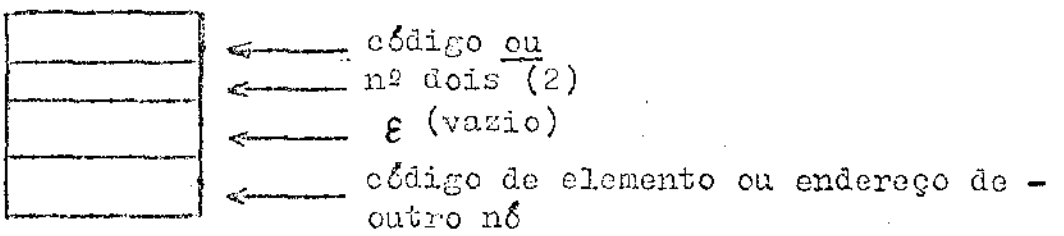


Fig 5.6

A árvore para armazenar a regra é construida interligando entre nós e atribuindo-se um endereço à raiz (correspondente à parte esquerda) armazenando-o numa tabela.

EXEMPLO 5.1 :- Para a regra :

$$P \rightarrow A + B + [\, B + C + A \;,\; (A + C)\,]$$

supondo que os códigos atribuidos aos símbolos são

P — -4

A — -3

B — -8

C — -9

a seguinte estrutura seria montada

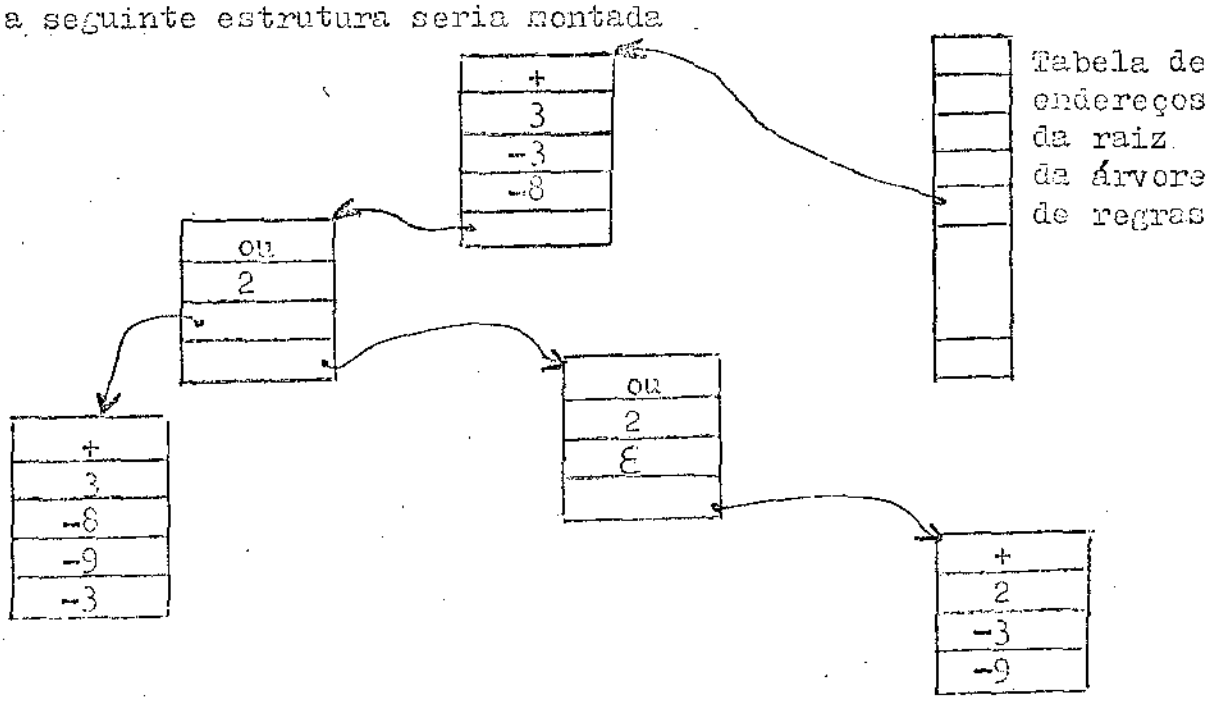


Fig. 5.7

5.2.2 - ANALISADOR

O comando ANALISE faz com que as frases que se seguem sejam analisadas e que todas as informaçoẽs sobre o seu reconhecimento sejam armazenados para posterior saída.

O analisador de frases da linguagem do linguista é do tipo "top-down".

Apesar desse método ser menos eficiente que o "botton-up" /14/, é particularmente adequado às nossas finalidades, por aceitar qualquer C- gramática, não impondo restriçoẽs de recursividade /17/, ambiguidade, etc.

Para minimizar as deficiências do método "top--drown", foi empregado um algoritmo baseado no fechamento transitivo /13/ e também tiveram que ser adotadas técnicas para tratar do problema da recursividade e ambiguidade.

5.2.2.1 - FECHAMENTO TRANSITIVO

O fechamento transitivo é calculado visando através de seu uso, diminuir a frequência do "back-track" (vide item 2.3.2) pelo fato da derivação a ser testada, em cada passo da análise, obedecer a um critério favorável de escolha : devemos verificar se o não terminal a ser substituido (parte esquerda da regra)pode gerar uma sequência que comece com o terminal no topo da cadeia de entrada, caso em que adotamos essa derivação.

O método de cálculo do fechamento transitivo, - R+, para uma relação R, esta totalmente desenvolvido utilizando-se o processo matricial /13/ e para tanto, monta-se uma matriz booleana para a relação R.

A relação R em que estamos interessados, é a relação para a regra cuja a parte direita "começa com". Assim para a regra :

P ⟶ A + C

diríamos que P R A

A notação utilizada pelo linguista para descrever uma regra pode originar uma ou mais em BNF (vide item 2.2.2) . Portanto, se dispusermos de todas as regras em notação BNF, é possível construir a matriz booleana para a relação R. Foi, para tanto , desenvolvido um algoritmo, programado utilizando o ND FORT (apêndice C) que "caminha" na árvore de regras obtendo as informações para montar a matriz.

Uma vez montada a matriz de relações R, é utilizado o algoritmo de Warshall /21/ para o cálculo do fechamento transitivo R+.

Todas as matrizes booleanas são armazenadas - "bit-a-bit" a fim de economizar espaço de memória.

5.2.2.2 - TRATAMENTO DA RECURSIVIDADE

Um método "top-down" pode analisar uma frase - dada, da direita para a esquerda ou da esquerda para a direita. Esta escolha é irrelevante.

Ao analisar uma recursão à direita percorrendo- a frase da esquerda para a direita, o número de passos efetuados pelo analisador é igual ao número de elementos que aparecem na frase; o mesmo acontece com relação à recursividade à esquerda analisada - da direita para a esquerda.

Tendo em vista essas considerações e acrescido- o fato das linguas naturais de origem latina (que constituirão a maior parte das gramáticas submetidas ao LL) serem comumente recursivas à direita, adotou-se o método de análise da esquerda para a direita.

Cuidados especiais devem ser tomados, porém na

análise de frases geradas por uma gramática recursiva à esquerda , do contrário o método "top-down" escolhido falhará totalmente, entrando em "loop". A solução adotada, nestes casos, é controlar o tamanho da pilha de análise, não permitindo que este ultrapasse o número de elementos que aparecem na frase. A este processo dá-se o nome de "Shaper". Portanto, a utilização do "shaper", quer no tratamento de recursividade, quer nas condições normais, vem sempre contribuir para tornar o método de análise mais geral e eficiente.

5.2.2.3. - TRATAMENTO DA AMBIGUIDADE

O método "top-down", baseia-se em tentativas efetuadas na escolha das regras para proceder a derivação com o objetivo de construir uma árvore de derivação. Analisar uma frase ambigua consiste em determinar todas as árvores existentes para a frase dada.

O procedimento para análise, neste caso, consiste em marcarmos os pontos onde foram tomadas as decisões com relação à derivação efetuada e qual foi a regra escolhida. Determinada uma árvore com um conjunto de escolhas bem sucedidas, voltamos àqueles pontos tentando outras substituições que levem a outras árvores possíveis.

5.2.2.4 - IMPLEMENTAÇÃO

A exaustiva programação de "tentativa-erro-retorno" envolvida no método "top-down" foi bastante simplificada pelo uso do FORTRAN- Não- Determinístico (ND FORT), cuja estrutura - permite tratar automaticamente o problema da ambiguidade, além de possibilitar a utilização dos recursos descritos nos itens anteriores. O programa para o analisador em ND FORT, encontra-se no apêndice B.

5.2.3 - ALGORITMO PARA CONSTRUÇÃO DA ÁRVORE DE DERIVAÇÃO

A saída fornecida pelo analisador, quando a frase é analisada com sucesso, consiste basicamente numa listagem das regras que foram sendo sucessivamente aplicadas.

Foi desenvolvido um algaritmo que, a partir dessa lista ordenada de regras, constrói a árvore de derivação e apresenta-a numa forma gráfica inteligível.

5.3 - PERFORMANCE

O sistema LL esta implementado em linguagem FORTRAN IV[1] para o PDP - 10 com 64 K e pode ser processado em outro computador, que aceite FORTRAN com 18 K palavras de memória disponíveis.

Cumpre-nos notar que não houve a preocupação relativa ao espaço de memória ocupado pelo sistema LL houve sim, por questões de segurança um super dimensionamento dos vetores e matrizes utilizados. Esta deficiência pode ser contornada utilizando-se os métodos de alocação de memória /19/, /20/, os quais controlam de maneira adequada os dimencionamentos efetuados e utilização do espaço disponível.

Quanto ao tempo de processamento, não nos é permitido comparações com outros sistemas devido a falta de referências neste sentido. Para a gramática com 8 regras e 6 elementos na frase de entrada, o tempo de CPU (CPU TIME) é da orden de 1.02 segundos, que pode ser considerado bastante adequado. Para uma gramática recursiva à esquerda (situação mais crítica para o analisador)

---

1 - Os programas escritos em MDFORT não traduzidos para FORTRAN IV por um processador especial /8/.

o tempo sofre um pequeno acréscimo mas ainda plenamente aceitável.

A recuperação de erros é bastante dificultada-pela própria estrutura modular do sistema, o que nos leva a considerar qualquer tipo de erro como sendo fatal.

CAPÍTULO 6

CONCLUSÕES

O sistema LL, como foi descrito, caracteriza-se por sua generalidade (aceitando C-gramáticas sem quaisquer restriçoes) e simplicidade de utilização, possibilitando um rápido aprendizado por pessoal não acostumado no uso de computadores.

O sistema implementado no PDP - 10 da Universidade Estadual de Campinas foi amplamente testado mostrou-se bastante confiável, consistindo uma útil ferramenta para o linguista.

Alguns melhoramentos podem ser introduzidos em futuras versoes do sistema, de maneira a torna-lo mais completo e versátil. Estas modificaçoes serão facilmente introduzidas ,devido à modularidade com que foi implementadoo sistema, possibilitando a incorporação de recursos adicionais ao analisador sintático e as rotinas semânticas.

Em particular dois melhoramentos se apresentam como os mais importantes e desejáveis :

a) - extensivo para S-gramáticas (sensíveis ao contexto)

b) - introdução de T-gramáticas (transformacionais).

Para o caso do item a deve-se acrescentar à C-gramática, regras que definem a dependência de contexto. Por exemplo a gramática do exemplo 1 do capítulo 4 gera a frase :

O MENINO VIROU O BARCO

por outro lado, gera também

O BARCO VIROU O MENINO

Esta última frase não faz sentido, dentro do âmbito de linguagens naturais embora sintaticamente correta, podendo ser de interesse - do linguista elimina-la do rol das frases gramaticais.

Frases deste tipo podem ser eliminadas das - linguagens por meio do acréscimo de restriçoes nas sequências de entrada (LEXICON) /2/ . Para o particular exemplo, as seguintes regras seriam incorporadas :

(MENINO , [+ N , + ARTIGO ——, + CONTÁVEL, + ANIMADO, + HUMANO])

(VIROU , [+ V , + ——NP , + [+ ANIMADO] AUX —— ARTIGO [+ CONCRETO] ....])

(BARCO , [+ N , + ARTIGO——, + CONCRETO , - HUMANO]).

Com relação ao item b, deve-se acrescentar regras [illegible], visando a permitir a geração de frases do tipo :

O BARCO FOI VIRADO PELO MENINO , a partir de

O MENINO VIROU O BARCO.

Trabalhos nesta área e adotando políticas muito semelhantes vem de ser bastante desenvolvido recentemente , tais como os descritos em /11/ e /15 /.

APÊNDICE A - BNF para geração do analisador sintático

```
$
BNF GRAM TER NTER AXIOM SEGRE LTER STRI1 LREG VNTER PVIR
REG PESQ PDIR MAIS ELEM LPAR LCOL LELEM VIRG2 STRI2 LPERG PERG
IMPRE ANALI ANA LSENT SENT LPAL PAL PVIRG

'$' 'VT' ':' ';' ',' 'VN' 'AXIOMA' 'REGRAS' '->' '+' ')' ']' '('
'[' 'IDENT' 'ANALISE' 'IMPRESSAO' 'DE'

!!!
BNF -> '$' GRAM '$' .
GRAM -> NTER TER AXIOM SEGRE LPERG .
NTER -> VNTER ':' LTER ';' #2 .
VNTER -> 'VN' #1 .
LTER -> LTER STRI1 / STRI1 .
STRI1 -> 'IDENT' #3 .
TER -> 'VT' ':' LTER ';' #4 .
AXIOM -> 'AXIOMA' ':' STRI2 ';' .
STRI2 -> 'IDENT' #5 .
SEGRE -> 'REGRAS' ':' LREG #17 .
LREG -> LREG REG / REG .
PVIR -> ';' #13 .
REG -> PESQ '->' PDIR PVIR .
PESQ -> 'IDENT' #6 .
PDIR -> PDIR MAIS ELEM / ELEM .
MAIS -> '+' .
ELEM -> 'IDENT' #7 / LPAR PDIR ')' #11 / LCOL LELEM ']' #12 .
LPAR -> '(' #8 .
LCOL -> '[' #9 .
LELEM -> PDIR / LELEM VIRG2 PDIR .
VIRG2 -> ',' #10 .
LPERG -> LPERG PERG / PERG .
PERG -> ANALI / IMPRE .
ANALI -> ANA ':' LSENT .
ANA -> 'ANALISE' #18 .
LSENT -> LSENT SENT / SENT .
SENT -> LPAL PVIRG / PVIRG .
LPAL -> LPAL PAL / PAL .
PAL -> 'IDENT' #14 .
PVIRG -> ';' #15 .
IMPRE -> 'IMPRESSAO' 'DE' 'REGRAS' ';' #16 .
$
```

APÊNDICE B – Programa em FORTRAN – NÃO – DETERMINISTICO (NDFORT) para implementação do analisador da gramática do linguista.

```
      COMMON /HASHI/HASHIN(0/1000),CODIGO,LIVRE,TABELA(-100/-1)
      COMMON /MEMORIA/MEM(500),MEMOR
      COMMON /SINAL/PARESQ,COLESQ,VIRGULA,MARCA,MAIS,OU,EPSILON,SUS
      COMMON /ENDERECO/PESQUE,ENDE(-100/-1)
      COMMON /SIMBOLOS/COULNTERMINAL,COULTERMINAL,AXIOMA
      COMMON /ANALISE/NELEM,VELEM(0/50),LEITURA
      DIMENSION PILHA(0/100),PILHAUX(50,2)
C=======DEFINICAO DE VARIAVEIS
      LEITURA=0
      SOMELEM=1
      LINHA=0
      TOPO=1
      PILHA(0)=0
      ESCOLHA=0
      PILHA(TOPO)=AXIOMA
C=======INICIO DO PROGRAMA EM NDFORT
      START
      IN ELEM
1000  T = PILHA(TOPO)
      IF (T.EQ.0) FAILURE
C=======TESTE SE O TOPO DA PILHA E TERMINAL
      IF (T.GE.COULNTERMINAL) GO TO 4000
      IF (T.EQ.EPSILON) GO TO 2000
      IF (T.EQ.MARCA) GO TO 3000
      IF (ELEM.NE.T) FAILURE
      IN ELEM
      SOMELEM = SOMELEM-1
2000  TOPO = TOPO-1 INV TOPO+1
      GO TO 1000
3000  IF (TOPO.EQ.1.AND.LEITURA.EQ.(NELEM+1)) SUCCESS
      TOPO = TOPO-1 INV TOPO+1
      GO TO 1000
4000  POSI = (IABS(T)-1)*IABS(COULTERMINAL-2)+IABS(ELEM)
      CALL OPERMA(POSI,VALOR,1,1)
      IF (VALOR.EQ.0) FAILURE
      OUT MARCA , T
      SOMELEM = SOMELEM-1
      PILHA(TOPO) = MARCA
      TAUX = ENDE(T)
C=======SUBSTITUIR O NAO TERMINAL NO TOPO DA PILHA PELA REGRA
4001  GO TO ( 5000 , 6000 ) , MEM(TAUX)
```

```
C=======MAIS
5000     K = TAUX+1+MEM(TAUX+1)
8000     IF (MEM(K).GT.0) GO TO 7000
         TOPO = TOPO+1
         PILHA(TOPO) = MEM(K)
         IF (MEM(K).EQ.EPSILON) GO TO 8003
         SOMELEM = SOMELEM+1
8003     IF (SOMELEM.GT.NELEM) FAILURE
         OUT MEM(K) ,
         IF (MEM(TAUX).EQ.OU) GO TO 8002
         K = K-1
8001     IF ((K-TAUX-1).GT.0) GO TO 8000
8002     IF (LINHA.EQ.0) GO TO 1000
         K = PILHAUX(LINHA,1)-1
         TAUX = PILHAUX(LINHA,2)
         LINHA = LINHA-1
         GO TO 8001
7000     IF (MEM(TAUX).EQ.OU) GO TO 7001
         LINHA = LINHA+1
         PILHAUX(LINHA,1) = K
         PILHAUX(LINHA,2) = TAUX
7001     TAUX = MEM(K)
         GO TO 4001
C=======OU
6000     ESCOLHA = CHOICE (MEM(TAUX+1))
         K = TAUX+ESCOLHA+1
         GO TO 8000
         END
```

APÊNDICE C - Programa em FORTRAN - NÃO - DETERMINISTICO (NDFORT) para implementação do cálculo da matriz booleana para relação R (parte direita da regra que "começa com").

```
      COMMON /HASHI/HASHIN(0/1000),CODIGO,LIVRE,TABELA(-100/-1)
      COMMON /MEMORIA/MEM(500),MEMOR
      COMMON /SINAL/PARESQ,COLESQ,VIRGULA,MARCA,MAIS,OU,EPSILON
      COMMON /ENDERECO/PESQUE,ENDE(-100/-1)
      COMMON /SIMBOLOS/COULNTERMINAL,COULTERMINAL
      DIMENSION PILHA(50,2)
C=======DEFINICAO DE VARIAVEIS
      LINHA=0
      ELEMVN=0
      ESCOLH=0
      NELEM=IABS(COULNTERMINAL)
C=======INICIO DO PROGRAMA EM NDFORT
      START
      ELEMVN = CHOICE (NELEM)
      PESQRE = -ELEMVN
      ENDREG = ENDE(PESQRE)
      IF (ENDREG.LE.0) FAILURE
C=======SAIDA DA PARTE ESQUERDA DA REGRA
      OUT PESQRE
9000  GO TO ( 2000 , 3000 ) , MEM(ENDREG)
2000  APONT = ENDREG+2
6000  IF (MEM(APONT).GT.0) GO TO 4000
C=======SAIDA DOS ELEMENTOS DA PARTE DIREITA DA REGRA
      OUT MEM(APONT)
      IF (MEM(ENDREG).EQ.OU) GO TO 5000
      APONT = APONT+1
7000  IF (APONT.LE.(ENDREG+MEM(ENDREG+1)+1)).GO TO 6000
5000  IF (LINHA.EQ.0) SUCCESS
      APONT = PILHA(LINHA,1)+1
      ENDREG = PILHA(LINHA,2)
      LINHA = LINHA-1
      GO TO 7000
4000  IF (MEM(ENDREG).EQ.OU) GO TO 8000
      LINHA = LINHA+1
      PILHA(LINHA,1) = APONT
      PILHA(LINHA,2) = ENDREG
8000  ENDREG = MEM(APONT)
      GO TO 9000
3000  ESCOLH = CHOICE (MEM(ENDREG+1))
      APONT = ENDREG+2+(MEM(ENDREG+1)-ESCOLH)
      GO TO 6000
      END
```

APÊNDICE D - Utilização do sistema LL para o exemplo 1 (gramática G1(S)) de modo conversacional.

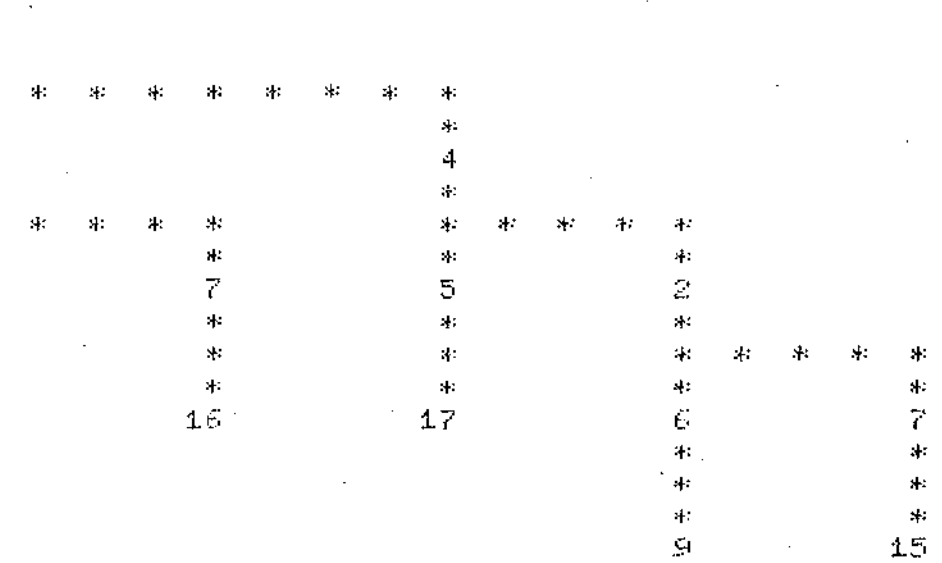
```
.RUN LL

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

IMPRIMA CONVE PARA CONVERSACIONAL
OU O NOME DO ARQUIVO
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

*CONVE

? $

? VN : S NP AUX VP V DET N M ;

? VT : O A UMA PODE DEVE SINCERIDADE MENINO BARCO VIRAR BUSCAR ;

? AXIOMA : S ;

? REGRAS : S -> NP + < AUX > + VP ;

? VP -> V + NP ;

? NP -> < DET > + N ;

? DET -> [ O , A , UMA ] ;

? AUX -> M ;

? M -> [ PODE , DEVE ] ;

? N -> [ SINCERIDADE , MENINO , BARCO ] ;

? V -> [ VIRAR , BUSCAR ] ;

? ANALISE : O MENINO PODE VIRAR O BARCO ;
```

```
==================
FRASE A SER ANALISADA

O MENINO PODE VIRAR O BARCO

? ARVORE ( 0 ) - REGRAS ( 1 ) - AMBAS ( 2 )
2


==================
REGRAS APLICADAS


S -> NP + AUX + VP
NP -> DET + N
DET -> O
N -> MENINO
AUX -> M
M -> PODE
VP -> V + NP
V -> VIRAR
NP -> DET + N
DET -> O
N -> BARCO

==================
ARVORE DE DERIVACAO


       1
       *
       * * * * * * * * * * * * *
       *               *       *
       2               3       4
       *               *       *
       * * * * *       *       * * * * *
       *       *       *       *       *
       6       7       8       5       2
       *       *       *       *       *
       *       *       *       *       * * * * *
       *       *       *       *       *       *
       9      15      12      17       6       7
                                       *       *
                                       *       *
                                       *       *
                                       9      16
```

```
? O MENINO PODE ;

====================
FRASE A SER ANALISADA

O MENINO PODE
###### A FRASE NAO PERTENCE A LINGUAGEM
       GERADA PELA GRAMATICA
```

```
? MENINO VIRAR O BARCO ;
====================
FRASE A SER ANALISADA

MENINO VIRAR O BARCO

? ARVORE ( 0 ) - REGRAS ( 1 ) - AMBAS ( 2 )
1

====================
REGRAS APLICADAS


S -> NP + VP
NP -> N
N -> MENINO
VP -> V + NP
V -> VIRAR
NP -> DET + N
DET -> O
N -> BARCO
```

```
? O BARCO VIRAR O MENINO ;

====================
FRASE A SER ANALISADA

O BARCO VIRAR O MENINO

? ARVORE ( 0 ) - REGRAS ( 1 ) - AMBAS ( 2 )
2


====================
REGRAS APLICADAS


S -> NP + VP
NP -> DET + N
DET -> O
N -> BARCO
VP -> V + NP
V -> VIRAR
NP -> DET + N
DET -> O
N -> MENINO


====================
ARVORE DE DERIVACAO
```



```
 ? $$

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

IMPRIMA CONVE PARA CONVERSACIONAL
OU O NOME DO ARQUIVO
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

*EXIT


CPU TIME: 3.42  ELAPSED TIME: 9:44.60
NO EXECUTION ERRORS DETECTED

EXIT
```

APÊNDICE E - Utilização do sistema LL para o exemplo 1 (gramática G1(S)) através do arquivo ANAE

```
. RUN LL


&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

IMPRIMA CONVE PARA CONVERSACIONAL
OU O NOME DO ARQUIVO
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

*ANAE

====================
FRASE A SER ANALISADA

MENINO PODE VIRAR O BARCO

? ARVORE ( 0 ) - REGRAS ( 1 ) - AMBAS ( 2 )
1


====================
REGRAS APLICADAS


S -> NP + AUX + VP
NP -> N
N -> MENINO
AUX -> M
M -> PODE
VP -> V + NP
V -> VIRAR
NP -> DET + N
DET -> O
N -> BARCO
====================
FRASE A SER ANALISADA

O MENINO PODE
###### A FRASE NAO PERTENCE A LINGUAGEM
       GERADA PELA GRAMATICA
```

```
====================
FRASE A SER ANALISADA

MENINO PODE VIRAR BARCO

? ARVORE ( 0 ) - REGRAS ( 1 ) - AMBAS ( 2 )
1


====================
REGRAS APLICADAS


S -> NP + AUX + VP
NP -> N
N -> MENINO
AUX -> M
M -> PODE
VP -> V + NP
V -> VIRAR
NP -> N
N -> BARCO
====================
FRASE A SER ANALISADA

MENINO VIRAR BARCO

? ARVORE ( 0 ) - REGRAS ( 1 ) - AMBAS ( 2 )
1


====================
REGRAS APLICADAS


S -> NP + VP
NP -> N
N -> MENINO
VP -> V + NP
V -> VIRAR
NP -> N
N -> BARCO
====================
FRASE A SER ANALISADA

O MENINO VIRAR BARCO

? ARVORE ( 0 ) - REGRAS ( 1 ) - AMBAS ( 2 )
1
```

```
====================
REGRAS APLICADAS


S -> NP + VP
NP -> DET + N
DET -> O
N -> MENINO
VP -> V + NP
V -> VIRAR
NP -> N
N -> BARCO
====================
FRASE A SER ANALISADA

MENINO VIRAR O BARCO

? ARVORE ( 0 ) - REGRAS ( 1 ) - AMBAS ( 2 )
1



====================
REGRAS APLICADAS


S -> NP + VP
NP -> N
N -> MENINO
VP -> V + NP
V -> VIRAR
NP -> DET + N
DET -> O
N -> BARCO


&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

IMPRIMA CONVE PARA CONVERSACIONAL
OU O NOME DO ARQUIVO
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

*EXIT


CPU TIME: 2.62  ELAPSED TIME: 1:35.30
NO EXECUTION ERRORS DETECTED

EXIT
```

APÊNDICE F - Utização do sistema LL para o exemplo 2 (gramática G2(S)) do modo conversacional.

```
 RUN LL


&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

IMPRIMA CONVE PARA CONVERSACIONAL
OU O NOME DO ARQUIVO
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

*CONVE

? $

? VN : S X ;

? VT : A ;

? AXIOMA : S ;

? REGRAS : S -> X ;

? X -> [ A + X , A ] ;

? ANALISE : A A A A A A ;
```

==========
FRASE A SER ANALISADA

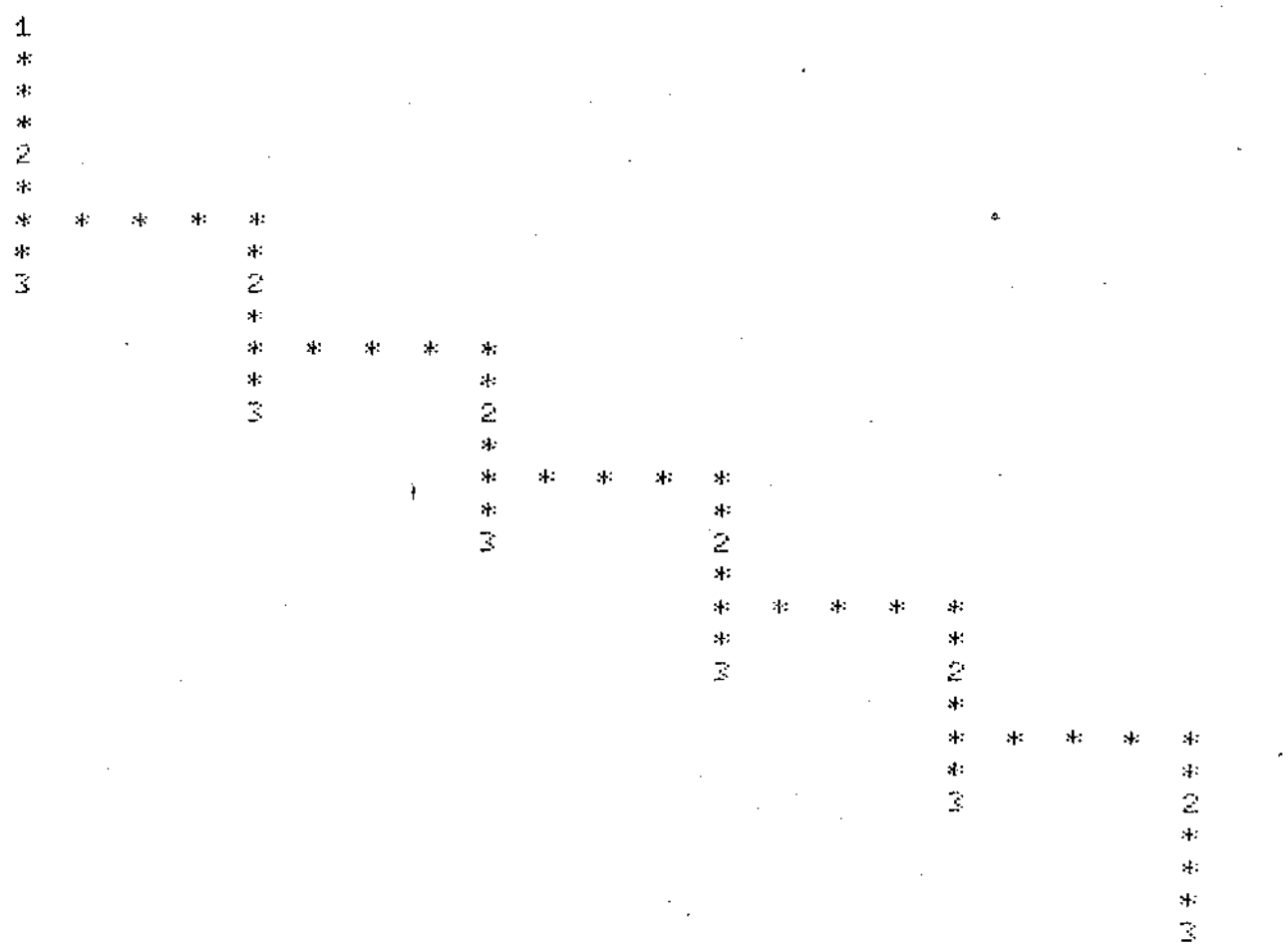
A A A A A A

? ARVORE (0) - REGRAS (1) - AMBAS (2)
2

==========
REGRAS APLICADAS

S -> X
X -> A + X
X -> A + X
X -> A + X
X -> A + X
X -> A + X
X -> A

==========
ARVORE DE DERIVACAO



```
? A

====================
FRASE A SER ANALISADA

A

? ARVORE ( 0 ) - REGRAS ( 1 ) - AMBAS ( 2 )
2


====================
REGRAS APLICADAS


S -> X
X -> A


====================
ARVORE DE DERIVACAO

                                    1
                                    *
                                    *
                                    *
                                    2
                                    *
                                    *
                                    *
                                    3

? $$


&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

IMPRIMA CONVE PARA CONVERSACIONAL
OU O NOME DO ARQUIVO
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

*EXIT


CPU TIME: 2.05  ELAPSED TIME: 4:9.40
NO EXECUTION ERRORS DETECTED

EXIT
```

APÊNDICE G - Utilização do sistema LL de modo conversacional mostrando uma condição de erro.

```
. RUN LL


&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

IMPRIMA CONVE PARA CONVERSACIONAL
OU O NOME DO ARQUIVO
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

*CONVE

? $

? VN : S Y X ;

? VT : A ;

? AXIOMA : S ;

? REGRAS : S -> Y ;

? Y -> X X ;

++++++ERRO NA LINHA NUMERO     6
      ACUSADO NA SINTAXE
Y -> X X ;
       ^

CPU TIME: 0.41  ELAPSED TIME: 1:24.55
NO EXECUTION ERRORS DETECTED

EXIT
```

REFERÊNCIAS BIBLIOGRAFICAS

/1/ BACH, Emmon. An introduction to transformational grammars. — New York, Holt /c1964/ 205p.

/2/ CHOMSKY, Noam. Aspects of the theory of syntax. Cambridge, M. I.T. Pr /c1965/ 251p.

/3/ - - - - - - . Language and mind. New York, Harcourt c/1968, - 1972/ 194p.

/4/ CHOMSKY, Noam. On certain formal properties of grammars. Informations and Control, New York, 2:137-167, 1959.

/5/ - - - - - - . Syntatic structure. New York, Humanities, 1957.

/6/ - - - - - - . Topics in theory of generative grammar. New York, Humanities, 1966.

/7/ COHEN, Jacques,et alli. A compiler generator : WALTHAM, Brandeis Univ., 1972.

/8/ - - - - - - . & CARTON, E. Non-deterministic Fortran. WALTHAM,- Brandeis Univ., 1973. 29p. /cópia xerox/.

/9/ DECSYSTEM 10;langu. handbook. 2 ed. Maryland, Digital /1972/ 388p. (Decsystem 10 Handbook Series).

/10/ DECSYSTEM 10; users handbook. 2 ed. Maryland, Digital, /1972/ 836p. (Decsystem 10 Handbook Series).

/11/ FRIEDMAN, Joyce. A computer system for transformational gra— mmar. Communications of the ACM, New York, 12(6):341-348,- 1969.

/12/ FLOYD, Robert W. Non-deterministic algorithms. Journal of the Association for Computing Machinery. New York, 14(4):636-644. 1968.

/13/ GRIES, David. Compiler construction for digital computers.New York, John Wiley /c1971/ 493p.

/14/ GRIFFITHS, T. V. & PETRICK, S. R. Top-down versus bottom-up -