TRADUTOR MICROLOBAN

SERGIO HENRIQUE CRIVOROT

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:

ESTEVAM GILBERTO DE SIMONE - Presidente

JOSÉ LUCAS MOURÃO RANGEL NETTO

MICHAEL ANTHONY STANTON

BEATRIZ ZAKIMI MIYASATO

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 1983

CRIVOROT, SERGIO HENRIQUE

TRADUTOR MICROLOBAN (Rio de Janeiro, 1983)

VII, 117 p. 29,7 cm (COPPE-UFRJ, M.Sc. Sistemas, 1983).

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

1. TRADUTOR MICROLOBAN. I.COPPE/UFRJ.II. Título (série)

A MEUS PAIS

AGRADECIMENTOS

- A meu orientador ESTEVAM que, não sei como, me convenceu a aceitar este trabalho. Agradeço por tudo que aprendí e pela sua atenção constante, mesmo após (infelizmente) ter-se desligado da COPPE.

- À ESSO BRASILEIRA DE PETRÓLEO S.A., que me permitiu cursar e concluir o Mestrado.

- À EXXON QUÍMICA S.A., em cujo computador o Tradutor MICROLOBAN foi desenvolvido.

- A meus amigos, pelo estímulo; especialmente à MARIA TERESA, pelo apoio e carinho.

- À RITA, minha artista gráfica favorita.

RESUMO

Microloban é um subconjunto da linguagem de operação de Banco de Dados Loban, que é suportado pelo Sistema de Gerência de Base de Dados Microban.

Este trabalho descreve o projeto e a implementação daqueles módulos da interface Loban que são do interesse da área de processadores de linguagens.

São descritos:

- o analisador léxico;
- o analisador sintático descendente, determinístico, com um símbolo de avanço construído a partir de uma gramática com lados direitos regulares através do método RRP LL(1). {5}
- um novo tradutor dirigido por sintaxe que utiliza uma gramática de saída gerando árvores binárias e que denominamos dendro-tradutor RRP.

São descritos ainda a maneira de usar esta implementação e o desenvolvimento de um programa que desenha as árvores geradas.

ABSTRACT

Microloban is a subset of the database operation language Loban, provided by the Microban Database Management System.

This thesis describes the design and development of the language processing modules:

- A common lexical analyser;
- A top-down deterministic parser with one symbol of look-ahead, generated from a regular right part grammar using RRP LL(1) method. {5}
- A specially developed syntax-directed translator with an output grammar producing binary trees - named RRP dendrum translator.

We describe how to use this language processor and a graphical output program which prints the translated trees.

ÍNDICE

I. INTRODUÇÃO

Microban é um sistema de gerência de base de dados interativo ou para processamento em lotes, autocontido e monousuário, e que suporta um subconjunto da linguagem de operação de banco de dados Loban [1]. Este subconjunto, denominado Microloban, é composto de comandos de definição, gerência e manipulação de base de dados, além dos de entrada e saída dos dados.

A Figura I.1, na página 2, mostra a arquitetura geral do sistema.

Este trabalho descreve a implementação do tradutor e seus módulos componentes. O objetivo principal foi discutir as idéias por trás da implementação, os caminhos que levaram à forma resultante, os aspectos teóricos envolvidos em cada módulo. Evitou-se entrar demasiadamente em detalhes do programa, muito mais adequados em um relatório técnico do que possivelmente ficariam em uma tese de mestrado.

O Capítulo II traz uma breve história do projeto Miniban, desde o seu início até o envolvimento do autor deste trabalho. Pretende-se esclarecer o leitor a respeito do significado desta implementação com relação ao projeto como um todo.

O Capítulo III descreve sucintamente as principais características da linguagem de operação de banco de dados, Loban, e da interface Microloban.

Na primeira parte do capítulo são definidos os conceitos introduzidos pela linguagem, isto é, os conceitos com os quais um usuário do sistema de gerência de banco de dados terá de se familiarizar.

Na segunda parte são listados alguns tipos pre-definidos em Microloban, comandos da linguagem, e algumas funções.

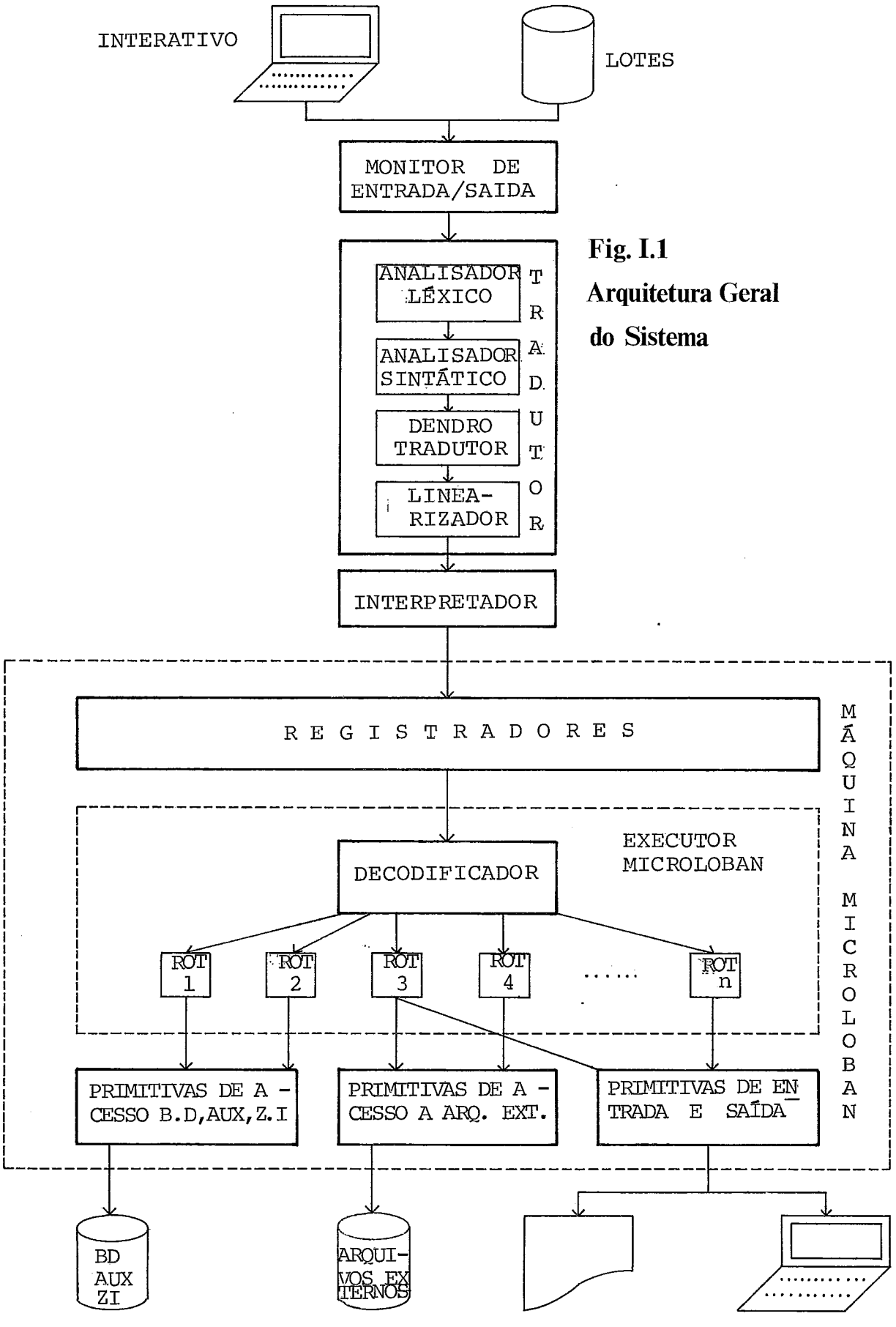


Fig. I.1
Arquitetura Geral do Sistema

No Capítulo IV é descrito o analisador léxico, modelado através de um transdutor finito, que traduz o programa fonte para uma sequência de unidades sintáticas que servirá de entrada para o analisador sintático.

O Capítulo V forma, juntamente com o Capítulo VI, o coração do trabalho.

No Capítulo V é estabelecida a base teórica necessária para a descrição do dendro-tradutor RRP. Os conceitos vão sendo introduzidos um a um, e no desenrolar do Capítulo vão se interrelacionando.

As gramáticas LL (1) são apresentadas. Apresenta-se então as gramáticas RRP, até se chegar às RRP LL(1), para as quais se constroi um analisador sintático. A seguir define-se árvores binárias e suas possíveis representações. Depois, o esquema de tradução dirigido pela sintaxe mostra a maneira de se realizar a geração de código simultaneamente com a análise sintática. Finalmente, todas as idéias são fundidas para o modelo de um SDTS, com gramáticas de entrada RRP LL(1), que emite árvores binárias na saída.

No Capítulo VI a teoria é colocada em prática nas seguintes etapas: a representação do Microloban através de uma gramática RRP-LL(1) e a implementação de seu analisador sintático; a definição do código intermediário a ser gerado para o interpretador Microloban, em forma de árvore binária linearizada; e a implementação do SDTS que o gera. Aborda-se ainda a estratégia de recuperação de erros utilizada.

O Capítulo VII mostra exemplos da utilização do tradutor, inclusive para fins didáticos, conjugado com o programa que desenha árvores, que permite uma representação gráfica do código intermediário gerado pelo tradutor Microloban.

Os demais capítulos são auto-explicativos.

II. HISTÓRICO

O Projeto Miniban - projeto de um sistema de banco de dados em minicomputador Nacional - começou a ser desenvolvido no ano de 1977 como resultado de um convênio entre o CNPQ, GMD da República Federal da Alemanha, Digibras e a UFRGS.

Miniban tem como objetivo a especificação de um sistema de banco de dados para um Minicomputador nacional, assim como o desenvolvimento de tecnologia nacional na área de banco de dados. A primeira etapa do projeto Miniban concentrou-se na especificação das estruturas de informação, as operações usando estas estruturas, assim como a definição da linguagem de operação de banco de dados Loban. A primeira implementação de um subconjunto Loban, denominado Sistema L, está sendo desenvolvido pela UFRGS. A partir de outubro de 1978 a COPPE/UFRJ iniciou sua participação neste projeto, dando origem ao projeto Miniban/COPPE.

A fase inicial do projeto Miniban/COPPE consistiu da definição detalhada de Loban, com a principal característica de separar nitidamente a etapa de definição da interface usuário/banco de dados, da etapa de realização ou implementação da mesma em um sistema portador.

Devido à extensão e extrema complexidade da interface assim definida, e ao desejo de adaptar Loban a um computador de pequeno-porte, a equipe do projeto Miniban/COPPE optou, como segunda etapa do projeto (denominado Microban), pelo desenvolvimento de um protótipo capaz de suportar um subconjunto Loban.

O sistema portador escolhido foi o Cobra-300 por ser uma máquina nacional e pela necessidade de desenvolvimento de software para sistemas deste porte.

O subconjunto Loban definido para a implementação do primeiro protótipo é denominado Microloban.

A primeira etapa da implementação deste protótipo é o desenvolvimento dos módulos de análise léxica e sintática e do tradutor para código intermediário.

Estas etapas são tarefas a serem executadas por pessoal que tenha como principais áreas de interesse linguagens de programação e o desenvolvimento de compiladores; e como área secundária, banco de dados.

Este é o caso do autor, que não pertence à equipe do projeto Miniban/COPPE. O trabalho, nominalmente, foi iniciado em 1981 mas todo este ano e a primeira metade de 1982 foram gastos ainda na definição do subconjunto a ser implementado. Pouco foi efetivamente feito no desenvolvimento do analisador e tradutor. O esforço maior foi, portanto, desenvolvido durante o segundo semestre de 1982 e o primeiro semestre de 1983.

Provavelmente, do ponto de vista de banco de dados, este trabalho padece de alguns males causados pelo seu desenvolvimento quase que totalmente em separado da equipe do projeto. Não é este o método que o autor recomenda a ninguém.

Como se trata de um conjunto de módulos e não da implementação total do Microloban, o autor teve o cuidado de escrever um programa de desenho de árvores que permite a fácil leitura da saída do tradutor, para auxiliar no desenvolvimento de etapas posteriores e, eventualmente, no ensino de Loban.

Espera-se que o tradutor venha efetivamente a ser utilizado por se tratar de uma pequena parte do grande esforço que deve ser desenvolvido para a efetiva geração de tecnologia nacional.

III. A LINGUAGEM LOBAN

Este capítulo tem como base a referência {2}, da qual o autor é um dos co-autores. Para um estudo detalhado do Loban, no entanto, recomenda-se {1}.

1. CARACTERÍSTICAS PRINCIPAIS DE LOBAN

A linguagem de operação de banco de dados (Loban) é uma linguagem autocontida que engloba comandos de definição, manipulação e gerência de dados, controle de acesso, providências de reconstrução, tratamento de erros e definição de transações, além de facilidades oferecidas pelas linguagens convencionais como definição de macros, comandos iterativos, condicionais e de entrada e saída de dados.

O desenvolvimento de Loban está baseado em conceitos IMC ("Information Management Concepts"), cuja principal característica é a distinção entre (construção de) informação (comumente denominado "valor" ou "conteúdo" de uma variável) e sua ocorrência dentro de um determinado contexto (conhecido como "nome de variável" ou "endereço de uma variável").

O conceito de construção de informação ou simplesmente construção, define qualquer "pedaço de informação" que possa ser referenciado por uma interface de gerência de base de dados. Uma construção é considerada elementar (ítem) se ela não é composta de outras construções, em oposição a um agregado que é formado por várias construções chamadas "componentes imediatos". Caso os componentes imediatos de um agregado possuam nomes, este é denominado nominação; caso contrário, ele é uma coleção.

Na abordagem relacional, a estrutura básica é a nominação de ítens - TUPLA. Um coletivo de Tuplas (com o mesmo conjunto de nomes) forma uma tabela relacional (conhecida na literatura como relação).

Na abordagem em redes é formada uma unidade de informação, ou seja, uma construção ligando um registro chamado "owner" a um conjunto de registros chamados "members". Em Loban este tipo de construção é denominado ligação, que é uma nominação de 2 elementos: uma tupla sob nome L e uma tabela relacional sob nome T. Um coletivo de ligações forma uma tabela ligacional ("set" em Codasyl).

Um arquivo em Loban é uma nominação sobre uma tabela e uma construção sob nome ficha, que contém informações como data de criação, atualização, etc. Dependendo do tipo de tabela componente, o arquivo será relacional ou ligacional.

Opcionalmente, Loban permite definir uma ou mais ordenações sobre a tabela de um arquivo. O agrupamento de arquivos constitui o acervo de trabalho (ACTRAB) que, juntamente com as construções usadas para providências de reconstrução, formam o acervo setorial (ACSET) que será o ambiente de trabalho para uma determinada aplicação. Cada acervo setorial tem associado a ele um nome, e o conjunto de todos os acervos setoriais forma o acervo total (base de dados), que é a construção mais abrangente suportada por Loban.

Para referenciar construções, Loban utiliza expressões denominadas endereços de pontos, que localizam construções dentro de um determinado contexto. Em geral, um endereço de ponto é uma seqüência de expressões booleanas, separadas pelo caracter ponto (.), que avaliadas num determinado nível de um agregado determinam os componentes imediatos que são selecionados (referenciados). Para que um componente imediato de um agregado seja selecionado, a expressão booleana correspondente deverá resultar no valor "verdadeiro".

Um recurso importante suportado por Loban é a marcação de pontos, isto é, depois de ter endereçado pontos na base de dados o usuário pode marcar estes pontos, atribuindo-lhes um nome, definindo "views" e "snapshots", cuja principal utilidade é

permitir referenciar estes pontos (em outros lugares) através do nome da marca, sem a necessidade de reavaliar o endereço de ponto que os selecionou.

O termo pretipo, em Loban, é usado para alguns tipos de cons - truções previamente definidos na linguagem, como por exemplo: real, inteiro, tupla, tabela, etc. Todo usuário tem a possibilidade de definir os seus "próprios" tipos de construções. Toda definição de um tipo de construção é feita através de ver - betes de coerência, que, em geral, constituem a definição das regras de consistência da base de dados.

Além dos verbetes de coerência, existem em Loban verbetes de usuário (para identificar os usuários do sistema), verbetes de acesso (para regulamentar os acessos a base de dados pelos diferentes usuários), entre outros. Todos estes verbetes são agrupados numa construção denominada folha (semelhante ao "schema" da Codasyl).

Todo usuário Loban possui uma área de trabalho denominada canal auxiliar, que funciona como uma base de dados particular e temporária, na qual ele pode armazenar, manipular e referenciar construções livremente.

Em geral, todos os resultados intermediários das operações são obtidos na zona intermediária, cujo conteúdo é perdido após a execução completa de um comando.

2. CARACTERÍSTICAS DA INTERFACE MICROLOBAN

Microloban é uma interface cuja especificação foi realizada sem a preocupação de como suas estruturas seriam representadas internamente na máquina escolhida para implementação. Esta separação deu maior liberdade de implementação, pois pode-se escolher dentre um maior número de alternativas, qual a solução mais viável quanto a realização no sistema portador.

Suas estruturas de informação fazem com que se tenha uma visão global da informação, e as operações sobre estas resolvem a maioria dos problemas de tratamento da informação na área de banco de dados. Tentou-se dar na sua definição a possibilidade de futuras extensões sem modificar suas funções originais.

Por ser uma interface poderosa, a proposta inicial é que sua definição e implementação se preste como uma ferramenta de ensino na área de banco de dados, e que funcione como um sistema de referência na comparação das diversas abordagens.

Microloban é uma linguagem autocontida, em português, que engloba diversas funções, dentre as quais estão as que descrevem a informação e seu relacionamento e as estruturas da base de dados (DDL), as funções de manipulação e uso da base de dados (DML), e funções normalmente realizadas por utilitários do SGBD, como reconstrução de acervos, modificações das definições dos dados, etc.

O principal critério adaptado na definição do subconjunto que compõe microloban foi o de reduzir a complexidade de Loban a um nível aceitável no sistema portador, porém mantendo a sua filosofia básica.

A comunicação do usuário com o sistema de banco de dados é feita utilizando uma instrução de trabalho, a qual corresponde a realização de um serviço. Uma instrução de trabalho é constituída de uma instrução de início, uma lista de instruções autôno -

mas e uma instrução de fim. Esta instrução de início determina se o processamento será interativo ou em lotes. Os dados a serem processados serão fornecidos ou junto com as instruções como anexos, ou separados em arquivos externos. Como resposta são fornecidos o resultado externo (ex.: relatórios), e o interno - que é guardado como uma base de dados caso tenham sido feitas alterações sobre a mesma. Serão também fornecidas mensagens operacionais padronizadas informando as ocorrências durante o processamento.

No que diz respeito a entrada/saída de dados, será utilizada uma única máscara (formato) padrão predefinida e várias regras de interpretação e representação, respectivamente.

Microloban permite alguns dos pretipos de Loban, entre os quais podemos citar:

- Pretipos atômicos: real, inteiro;

- Pretipos agregados básicos: numeração de caracteres, data, hora, tupla, ligação, coleção de ítens, tabelas relacionais e ligacionais;

- Folha, que conterá:
  - Descrição dos tipos de construção que compõem a base de dados (verbetes de coerência)
  - Autorizações permitidas (verbetes de acesso)
  - Identificação de usuários (verbete de usuário)
  - Procedimentos pre-definidos (verbete de texto fonte)

- Arquivo, composto de um tipo de construção padrão chamado ficha e uma tabela. Sendo assim, existem arquivos relacionais e ligacionais. Uma restrição feita em Microloban é a eliminação das relações de ordem tanto dos arquivos quanto das ligações.

- Ficha, contendo informações referentes ao arquivo ou acervo - setorial tais como: data de criação, data da última atualização, etc.

- ACTRAB (acervo de trabalho)

- ACSET (acervo setorial)

- ACTOT (acervo total)

Microloban permite os seguintes comandos:

- Comandos de gerência, onde o usuário tem a possibilidade de criar e abolir tanto acervos setoriais quanto arquivos.

- Comandos de manipulação, que permitem incluir, excluir e substituir construções tanto na base de dados quanto no canal auxiliar.

- Comandos de controle de fluxo, que permitem a execução repetitiva e/ou condicional . O comando iterativo permite a especificação de uma ordenação temporária sobre o conjunto de pontos a serem processados.

- Comandos de alocação de recursos com os quais o usuário informa a base de dados a ser usada, assim como os dispositivos de entrada/saída necessários. Além de definir qual a base de dados requisitada, o usuário deverá especificar a área de dados a ser usada (protegida) e para que tipo de acesso.

- Comandos de marcação, que permitem a definição de "snapshots", e não de vistas como é o caso de Loban.

- Comandos de reconstrução. São permitidos dois níveis de reconstrução: de sessão (de tipo regressiva) e de comandos dentro de uma sessão (tanto progressiva quanto regressiva).

- Comando de saída que permite representar construções no meio externo.

- Comando de definição de transação, que permite representar - conjunto de comandos cuja execução será considerada como uma unidade de processamento. No início da transação são es - pecificadas as ações que serão executadas quando da ocorrên - cia de êrros de execução.

Além dos comandos acima descritos temos as expressões que quando executadas geram construções na zóna intermediária (área de trabalho). Estas expressões não têm restrições quanto ao nível de embutimento e englobam as seguintes funções:

- Entrada de dados
- Operações aritméticas
- Operações da álgebra relacional
- Operações do cálculo relacional
- Operações sobre tabelas relacionais e ligacionais

Microloban não permite operações sobre cadeias de caracteres, - produto e concatenação cartesiana.

Por último, temos as expressões que permitem o endereçamento de pontos do acervo ou canal auxiliar, e campos dos volumes de entrada e saída.

IV. ANÁLISE LÉXICA

O desenho de um analisador léxico eficiente é tarefa simples. Farta teoria a respeito pode ser encontrada em {6} e {8}, que são as referências para todo o capítulo. A notação usada é a de {6}.

A função do analisador léxico é grupar seqüências de caracteres terminais em entidades sintáticas primitivas, conhecidas como - "Tokens". O que vai constituir ou não uma entidade sintática em uma dada implementação, embora se trate de uma decisão do projetista, é fundamentalmente influenciado pela especificação da linguagem.

A cada seqüência de símbolos terminais grupados, associa-se uma estrutura léxica consistindo de um par da forma (tipo, valor).O primeiro componente é um tipo de entidade sintática, tal como "identificador", e o segundo componente fornece informações que individualizam um elemento dentro do conjunto de seqüências de símbolos que pertencem ao mesmo tipo.

O primeiro componente do par é usado pelo analisador sintático, enquanto que o segundo é usado durante a fase de geração de código.

Assim, o analisador léxico é um tradutor cuja entrada é uma seqüência de símbolos representando o programa fonte e cuja saída é uma seqüência de entidades sintáticas primitivas. Esta saída forma a entrada do analisador sintático.

A melhor representação para as entidades sintáticas reconhecidas pelo analisador léxico é em forma de expressão regular.

Uma expressão regular (ER) e o conjunto de seqüências sobre o alfabeto Σ que denota são definidas como:

(1) $\emptyset$ é uma ER e denota o conjunto vazio;
(2) a pertencente a Σ é uma ER e denota o conjunto {a} ;
(3) Se p e q são ER's denotando os conjuntos P e Q, então:
    (p|q) é ER e denota P $\cup$ Q;
    (pq) é ER e denota PQ ;
    (p)* é ER e denota P*;
(4) Nada mais é ER.

Os parênteses redundantes podem ser removidos obedecida a precedência: fechamento, alternação, concatenação.

São aceitas também abreviaturas, com a mesma precedência que o fechamento:
(1) p? denotando P $\cup$ {ε};
(2) p+ denotando PP*;
(3) p£q denotando P(QP) *.

Na página 15, a figura IV.1 mostra algumas das entidades sintáticas reconhecidas em Microloban. São mostrados: a representação externa em forma de expressão regular e o par associado (tipo de entidade, valor). A tabela de tradução completa com todas as entidades sintáticas está no apêndice 1.

A representação externa está em forma de expressão regular sobre o alfabeto formado por todos os caracteres válidos no Cobra 300:

Σ ={"A", "B", "C", ..."Z", "0", "1", ..., "9", ";", ":", ...}

Algumas simplificações foram adotadas:
l = "A" | "B" | "C" | ... | "Z"
d = "0" | "1" | "2" | ... | "9"

As outras entidades com a mesma lei de formação que os identificadores são as palavras reservadas do Microloban, isto é, identificadores com significado pre-estabelecido.

FIGURA IV.1 - Parte da tabela de tradução do analisador léxico

REPRESENTAÇÃO	TIPO	VALOR
";"	PONTO-E-VÍRGULA	
","	VÍRGULA	
"."	PONTO	
"<"	MENOR	
">"	MAIOR	
":"	DOIS-PONTOS	
")"	FECHA-PARENTESES	
"("	ABRE-PARENTESES	
"<>"	DIFERENTE	
"<="	MENOR-OU-IGUAL	
">="	MAIOR-OU-IGUAL	
":="	ATRIBUIÇÃO	
$l(l/d/"\_")^*$	IDENTIFICADOR	REPRESENT.EXTERNA
d+	INTEIRO	REPRESENT.EXTERNA
d+ "," d+	REAL	REPRESENT.EXTERNA
dd "." dd "." $(dd)^?$ dd	DATA	REPRESENT.EXTERNA
dd ":" dd $(":" dd)^?$	HORA	REPRESENT.EXTERNA
ASPAS Σ* ASPAS	NUMCAR	REPRESENT.EXTERNA
"A"	A	
"C"	C	
"AO"	AO	
"CC"	CC	
"PC"	PC	
"COM"	COM	
"REC"	REC	
"SOBRE"	SOBRE	
"ALTERAR"	ALTERAR	
"EXCLUIR"	EXCLUIR	
"COLITENS"	COLITENS	
"COMPILAR"	COMPILAR	
"GERENCIAR"	GERENCIAR	
"REPRESENTAR"	REPRESENTAR	
"ALFANUMÉRICO"	ALFANUMÉRICO	
"TELEIMPRESSORA"	TELEIMPRESSORA	

A decisão do que é um token depende não só da linguagem, mas também do projetista do analisador. Por exemplo, poder-se-ia ter optado por não considerar a existência de um token do tipo literal data, mas sim considerar uma seqüência de cinco tokens: inteiro - ponto-inteiro-ponto-inteiro. Simplificações como esta levariam a um analisador léxico mais simples, mas ao custo de transferir o reconhecimento de um literal-data para a fase de análise sintática. Este enfoque, no entanto, é bastante discutível. Foi seguida a orientação segundo a qual é melhor complicar-se o analisador léxico, para simplificar o analisador sintático. A lista de "tokens", mostrada acima, é aquela que julgou-se estar mais próxima da estrutura do Loban, como definida pelo usuário.

Sempre que o analisador sintático precisa de uma nova entidade sintática, o analisador léxico é chamado, sendo, portanto, uma subrotina do analisador sintático. O analisador léxico só retornará o controle ao analisador sintático (com exceção do caso de fim de arquivo) após reconhecer uma entidade sintática válida para seu uso. Caso contrário continuará analisando o registro lido, enquanto for necessário. O analisador léxico escolherá sempre a mais longa entidade sintática possível.

Duas definições são importantes para a concepção do analisador léxico: automato finito determinístico e transdutor finito.

Automato finito não determinístico é uma 5-Tupla

$$M = (Q, \Sigma, \delta, q_0, F)$$

onde:

(1) Q é um conjunto finito de estados;

(2) Σ é um conjunto finito de símbolos;

(3) δ, a função de transição,

$$\delta: (Q \times \Sigma) \rightarrow Q^*$$

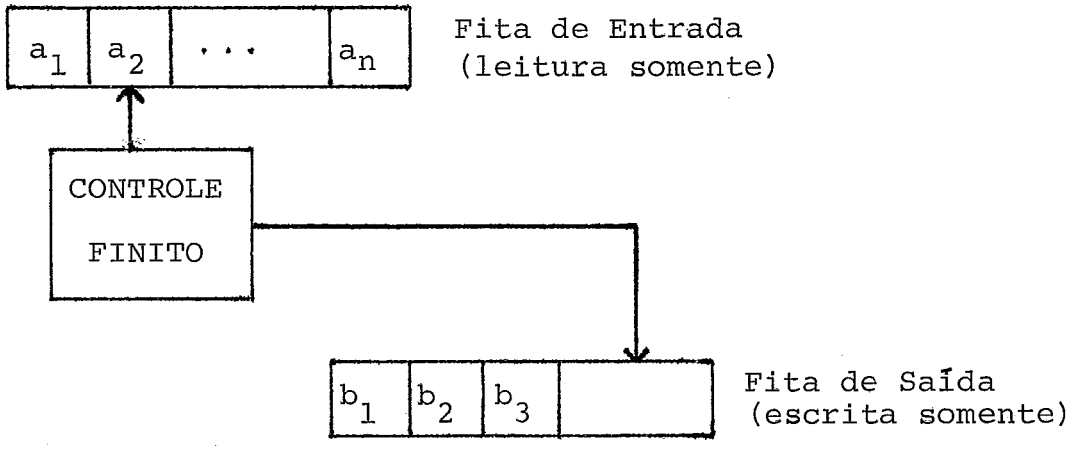
(4) q_0 é o estado inicial;

(5) F, que é um subconjunto de Q, é o conjunto de estados finais.

Se δ mapeia $(Q \times \Sigma)$ em Q diremos que o autômato é determinístico - (AFD).

Como já foi exposto, o analisador léxico é um tradutor, que traduz o programa fonte para uma sequência de entidades sintá - ticas.

O tradutor mais simples é o transdutor finito, usado por nós.Um transdutor é composto por um reconhecedor acoplado a um trans - missor, que emite um conjunto de símbolos de saída a cada movi- mento feito, podendo este conjunto ser vazio. O transdutor fi - nito é obtido tomando-se um autômato finito e permitindo a má - quina emitir um conjunto de símbolos de saída em cada movimen - to. A Figura IV.2 apresenta o modelo de um transdutor finito.



Um transdutor finito é uma 6-Tupla $(Q, \Sigma, \Delta, \delta, q_0, F)$ onde,

(1) Q é um conjunto finito de estados

(2) Σ é um alfabeto de entrada

(3) Δ é um alfabeto de saída

(4) δ é uma função de transição

$$\delta: (Q \times \Sigma) \rightarrow \mathcal{P}(Q \times \Delta^*)$$

(5) q_0 pertence a Q, sendo o estado inicial

(6) F está contido em Q, sendo o conjunto de estados finais

A análise léxica é dita direta quando, dada uma seqüência de símbolos de entrada e um ponteiro para aquela seqüência, o analisador determina a qual tipo de entidade sintática pertence o grupo de símbolos imediatamente a direita do lugar apontado e move o ponteiro para a direita daquele grupo de símbolos.

A maneira mais eficiente de se implementar um analisador léxico direto é através da busca em paralelo, pois a cada símbolo lido o número de possibilidades decresce rapidamente.

A estratégia de desenho adotada foi a seguinte:

Para cada tipo de entidade sintática foi desenhado um automato finito que a reconhecesse. Todos os automatos foram então combinados em um só, que foi tornado determinístico e mínimo.

A partir do automato finito determinístico combinado, obteve-se um transdutor finito simples que emite na saída o tipo de entidade sintática reconhecida e, eventualmente, alguma informação particular sobre seu valor.

Cada estado do automato representa estados de vários dos automatos componentes. Quando o automato combinado entra num estado que contem um estado final de um dos automatos componentes, e nenhum outro estado, ele para e emite o nome da entidade sintática reconhecida, se não houver novas transições possíveis.

O automato finito determinístico combinado, representado pelo analisador léxico está na Figura IV.3, na página 19.

Um problema ocorre no caso dos identificadores e das palavras reservadas. Devido a grande quantidade de identificadores possíveis, usa-se uma definição simplificada (letra seguida de qualquer combinação de letras, digitos e travessões) para o transdutor.

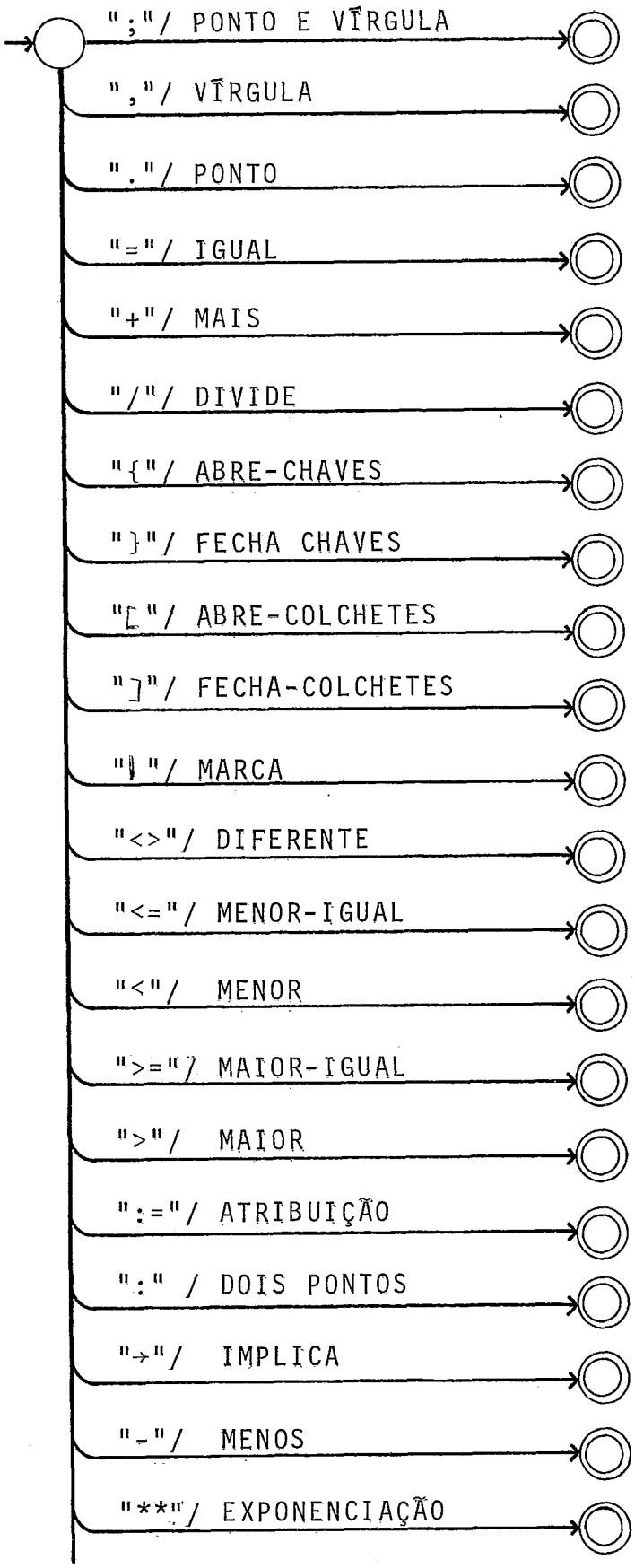
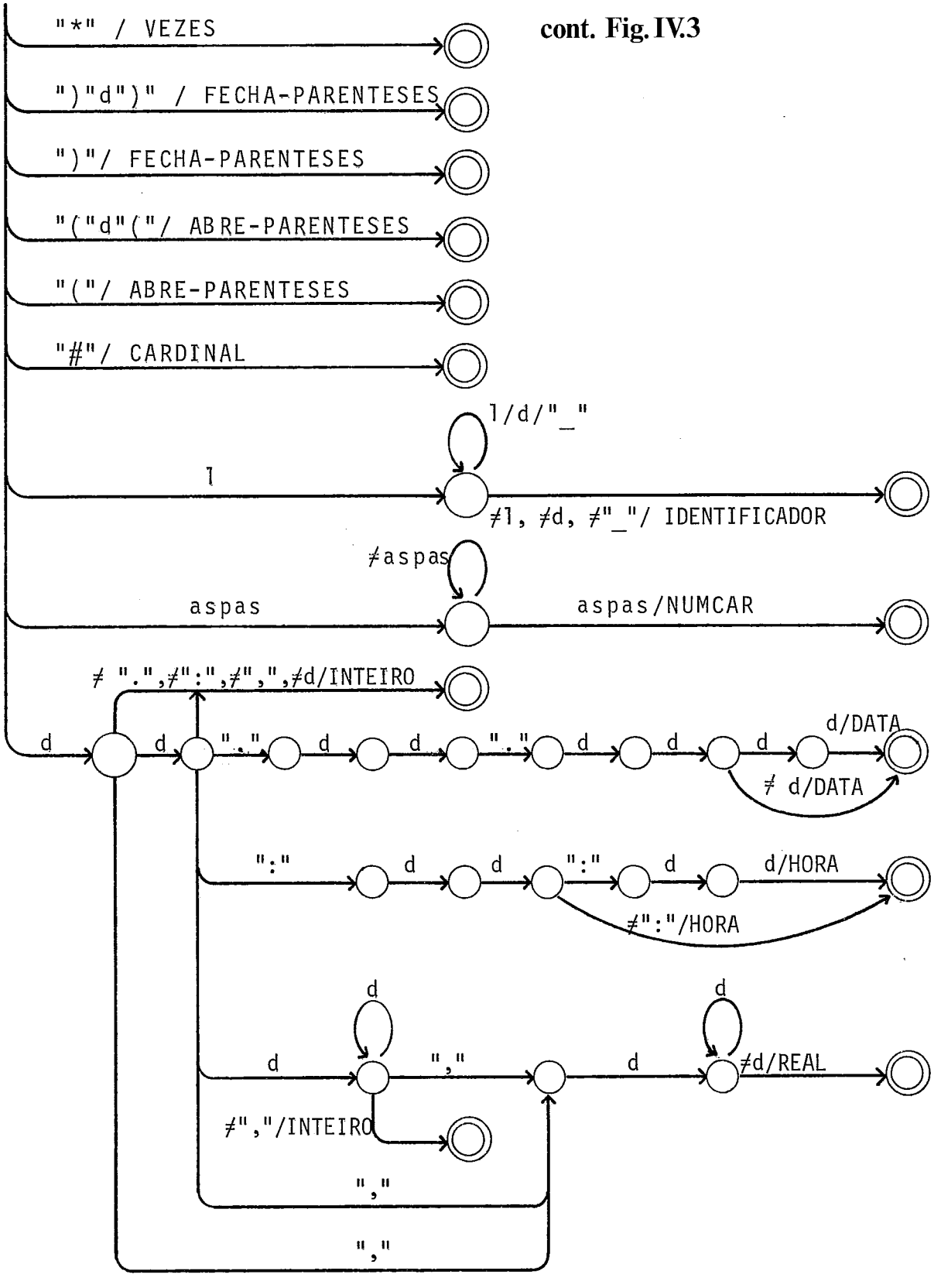


Fig. IV.3

Autômato Finito


"*" / VEZES
")"d")" / FECHA-PARENTESES
")"/ FECHA-PARENTESES
"("d"("/ ABRE-PARENTESES
"("/ ABRE-PARENTESES
"#"/ CARDINAL
l/d/"_"
l
≠l, ≠d, ≠"_"/ IDENTIFICADOR
≠aspas
aspas
aspas/NUMCAR
≠ ".",≠":",≠",",≠d/INTEIRO
d
d/DATA
≠ d/DATA
":"
d/HORA
≠":"/HORA
≠d/REAL
≠","/INTEIRO
","
","

cont. Fig. IV.3

Quando o transdutor reconhece um identificador, é executado um procedimento especial para determinar se ele pertence ao conjunto pre-definido, representado sob a forma de uma tabela. Caso isto ocorra, trata-se de uma palavra reservada, e não de um identificador.

A tabela de palavras reservadas é representada na memória como uma cadeia contínua de caracteres a partir da posição denominada "Tabreserva", ocupando 1130 bytes. As palavras reservadas lá são colocadas em ordem ascendente de comprimento e em ordem alfabética. Deste modo, temos todas as palavras de comprimento 1, depois todas as de comprimento 2, etc. A última palavra de cada grupo começa com um câracter inválido em palavras, o caracter "}", para indicar o final do grupo de palavras daquele comprimento.

Abaixo vai descrito o algorítmo de busca na tabela de palavras reservadas:

Passo 1: Inicial := função (comprim do ident, letra inicial)

Passo 2: Vezes := 0

Passo 3: Posição := tabreserva + inicial
+ (vezes * comprim do ident)

Passo 4: Compara caracter a caracter o identificador sendo analisado com o conteúdo das posições de memória
{Posição:Posição + Comprim - 1}

Passo 5: Se há igualdade, gera o token correspondente a palavra reservada:
Token:= função (posição, comprim do ident)

Passo 6: Se não há igualdade e a letra inicial do identificador sendo buscado é menor ou igual a letra em{posição}, então incrementa "vezes" e vai para o passo 3.

Passo 7: Se não há igualdade e a letra inicial do identificador sendo buscado é maior que a letra em{posição}, então o identificador em questão não é uma palavra reservada.

O algoritmo descrito acima foi elaborado visando-se um bom aproveitamento na memória, aspecto mais crítico desta implementação. Cada palavra utiliza apenas o seu comprimento real, sendo perdido apenas o espaço ocupado pela palavra fictícia colocada no final de cada grupo. Deste modo, 963 dos 1110 caracteres da tabela são efetivamente aproveitados, isto é, 87%.

As funções mencionadas nos Passos 1 e 5 são algebrismos bastante específicos, definidos a partir do conjunto de palavras reservadas no Loban, visando um bom aproveitamento da memória, como explicado acima. O aspecto lógico, que é fundamental, é explicado em detalhe no algorítmo.

Não há necessidade de manter-se uma tabela de identificadores. Toda a análise semantica estática será realizada pelo interpretador. O tradutor se limitará a passar para o interpretador uma descrição de cada identificador contido no programa lido, sem guardar nenhum registro a respeito.

V. ANÁLISE SINTÁTICA / GERAÇÃO DE CÓDIGO. DISCUSSÃO TEÓRICA

Neste Capítulo, com exceção da Seção 3, utiliza-se a notação e os conceitos básicos de teoria de conjuntos e teoria de linguagens desenvolvidos em {6}.

Na Seção 1 alguns conceitos bastante básicos são apresentados, tais como o de grâmatica, somente para um melhor encadeamento de idéias, sem que se pretendesse cobrir todo o alcance de um curso sobre linguagens

O estudo sobre gramáticas com lados direitos regulares e suas representações, desenvolvido na Seção 2, teve como referências {3} e {5}.

Maiores detalhes sobre o assunto abordado na Seção 3, Árvores Binárias, podem ser encontrados em {11}, cuja notação foi utilizada.

1. GRAMÁTICAS LL(1)

Um alfabeto é qualquer conjunto de símbolos.

Define-se uma seqüência de símbolos sobre um alfabeto Σ da seguinte maneira:

(1) ε é uma seqüência sobre Σ.

(2) Se x é uma seqüência sobre Σ e a está em Σ, então xa é uma seqüência sobre Σ.

(3) Nada mais é seqüência.

Uma linguagem sobre um alfabeto Σ é sempre um subconjunto de Σ^*. Linguagens de programação tais como FORTRAN e LOBAN estão obviamente incluídas nesta definição.

Uma linguagem composta de um número finito de seqüências de símbolos pode ser representada através de uma lista de todas essas seqüências. Para uma linguagem composta de um número infinito de seqüências outro método de representação necessariamente deve ser procurado.

Há diversos métodos de representação que preenchem este requisito. Usaremos um método generativo, chamado gramática. Cada sentença da linguagem pode ser construída através de métodos bem definidos, usando as regras (produções) da gramática. Representação através de gramáticas simplifica a análise sintática e a tradução, por causa da estrutura transmitida às sentenças da linguagem pela gramática.

Uma gramática é um sistema formal para se definir uma linguagem, bem como um dispositivo para dar às sentenças de linguagem uma estrutura útil.

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

(1) N é um conjunto finito de símbolos não-terminais (algumas vezes chamados de variáveis ou categorias sintáticas).

(2) Σ é um conjunto finito de símbolos terminais, disjunto de N.

(3) P é subconjunto finito de N x $(N \cup \Sigma)^*$
Um elemento (α,β) de P será escrito na forma $\alpha \rightarrow \beta$ e será chamado uma produção.

(4) S é um símbolo particular em N chamado símbolo inicial.

Uma gramática da forma descrita é dita livre de contexto se cada produção de P é da forma $A \rightarrow \alpha$, onde A pertence a N e α pertence a $(N \cup \Sigma)^*$. Outros tipos de gramáticas com diferentes definições para P existem, mas não são relevantes neste estudo.

Como mencionado anteriormente, a saída do analisador léxico é uma seqüência de pares (tipo de entidade sintática, valor). Esta seqüência forma a entrada do analisador sintático, que analisa somente os primeiros componentes dos pares - os tipos. A informação sobre cada um - segundo componente - é usada mais tarde, na geração de código. Portanto, os primeiros componentes de pares são os terminais da gramática.

A análise sintática ou "parsing" é o processo em que a seqüência de pares é examinada para determinar-se se ela obedece certas convenções estruturais explícitas na definição sintática da linguagem, isto é, se ela pertence à linguagem.

A partir de um conjunto de regras sintáticas é possível construir-se automaticamente analisadores sintáticos, ou "parsers" que garantirão que um programa fonte obedece à estrutura sintática definida por estas regras sintáticas.

Restringindo-se à classe de gramáticas em estudo, é possível construir-se analisadores sintáticos mais eficientes. Discutiremos o caso de algorítmos de análise sintática caracterizados pelo fato de que a seqüência de entrada é lida somente uma vez da esquerda para a direita e o processo de análise é completamente determinístico.

Na verdade, estamos restringindo a classe de gramáticas livres de contexto de modo que possamos construir um "parser" esquerdo determinístico para a gramática sendo considerada.

Diversos conceitos utilizados a seguir serão considerados já definidos, encontrando-se no estudo de teoria de linguagens realizado em {6}. São êles:

- Derivação (=>)
- Derivação esquerda ($\underset{lm}{=>}$)
- "Parse"
- "Parsing" preditivo
- Forma sentencial (α, β, γ, ...)

Para uma gramática livre de contexto G = (N,Σ,P,S), define-se - FIRST (α) = $\{x \mid \alpha \overset{*}{\underset{lm}{=>}} x\beta\}$, onde lm significa derivação esquerda. Isto é, FIRST (α) consiste de todos os símbolos terminais iniciais das seqüências de símbolos que podem ser derivadas a partir de α.

Diz-se que a gramática G é LL(1), se sempre que há duas derivações esquerdas:

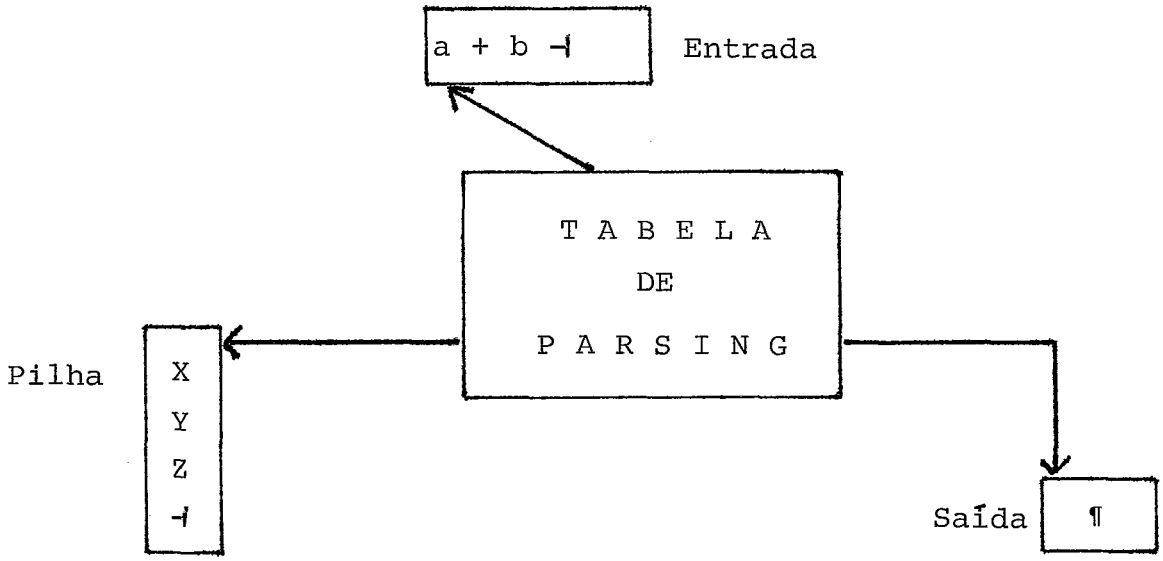
(1) $S \overset{*}{=>} wA\alpha \Rightarrow w\beta\alpha \overset{*}{\Rightarrow} wx$; e

(2) $S \overset{*}{=>} wA\alpha => w\gamma\alpha \overset{*}{=>} wy$,

tais que FIRST (x) = FIRST (y), então segue-se que $\beta = \gamma$.

Menos formalmente, G é LL(1) se dada uma seqüência de símbolos - $wA\alpha$ em $(N \cup \Sigma)^*$ e o primeiro símbolo terminal derivado de $A\alpha$, existe no máximo uma produção que pode ser aplicada a A para produzir uma derivação de qualquer seqüência de terminais começando - por w seguido por este terminal.

É possível efetuar-se o "parse" de gramáticas LL(1) de maneira - muito conveniente através de um algorítmo de parsing preditivo de um símbolo, usando-se uma fita de entrada, uma pilha e uma fita-de saída. O algorítmo preditivo de um símbolo tenta achar uma derivação esquerda da seqüência colocada em sua fita de entrada.



A fita de entrada contém a seqüência de entrada a ser analisada, seguida de ⊣ , o marcador de final. Ela é lida por uma cabeça capaz de ler o próximo símbolo, dito o símbolo de avanço.

A pilha contém uma seqüência de símbolos da gramática, precedidos de ⊣ . A tabela de controle do analisador é uma tabela bidimensional M(A,a), onde A é um não-terminal, e a é um terminal ou o símbolo ⊣ .

O analisador sintático é controlado por um programa que funciona da maneira descrita a seguir: o programa determina X, o símbolo no topo da pilha, e a, o símbolo atualmente na entrada. Estes dois símbolos determinam a ação do analisador. Há 3 possibilidades:

(1) Se X = a = ⊣, o analisador para e anuncia que a análise foi concluída com sucesso.

(2) Se X = a ≠ ⊣, o analisador desempilha X e avança o ponteiro da entrada para o próximo símbolo da entrada.

(3) Se X é um não-terminal, o programa consulta a entrada da tabela M(X,a). Esta entrada será ou uma produção da gramática, cujo lado esquerdo é o não-terminal X, ou será uma entrada de erro. Se M(X,a) = (X → UVW), o analisador substitue no topo da pilha por WVU, com U no topo. Se M(X,a) = erro, uma rotina de recuperação de erros pode ser chamada.

(4) Em qualquer outra situação acusará erro.

Para descrevermos um algorítmo de construção da tabela de controle para análise sintática preditiva, falta-nos ainda uma ferramenta.

Seja G=(N,Σ,P,S) uma gramática livre de contexto. Define-se FOLLOW (β), sendo β pertencente a (NUΣ)*, como sendo o conjunto {w | S $\stackrel{*}{\Rightarrow}$ αβγ e w pertence a FIRST(γ)}.

Isto é, FOLLOW (A) inclue o conjunto de símbolos terminais que podem ocorrer imediatamente a direita de A em qualquer forma sentencial.

A idéia por trás do algorítmo de construção da tabela é simples: seja A→α uma produção com a pertencente a FIRST (α), então sempre que o analisador tem A no topo da pilha com a como símbolo na entrada, o analisador expande A por α. A única complicação ocorre quando α = ε ou α$\stackrel{*}{\Rightarrow}$ε. Neste caso deve-se expandir também A por α se o símbolo na entrada pertence ao FOLLOW (A), ou se o ⊣ na entrada foi atingido e ⊣ pertence ao FOLLOW(A).

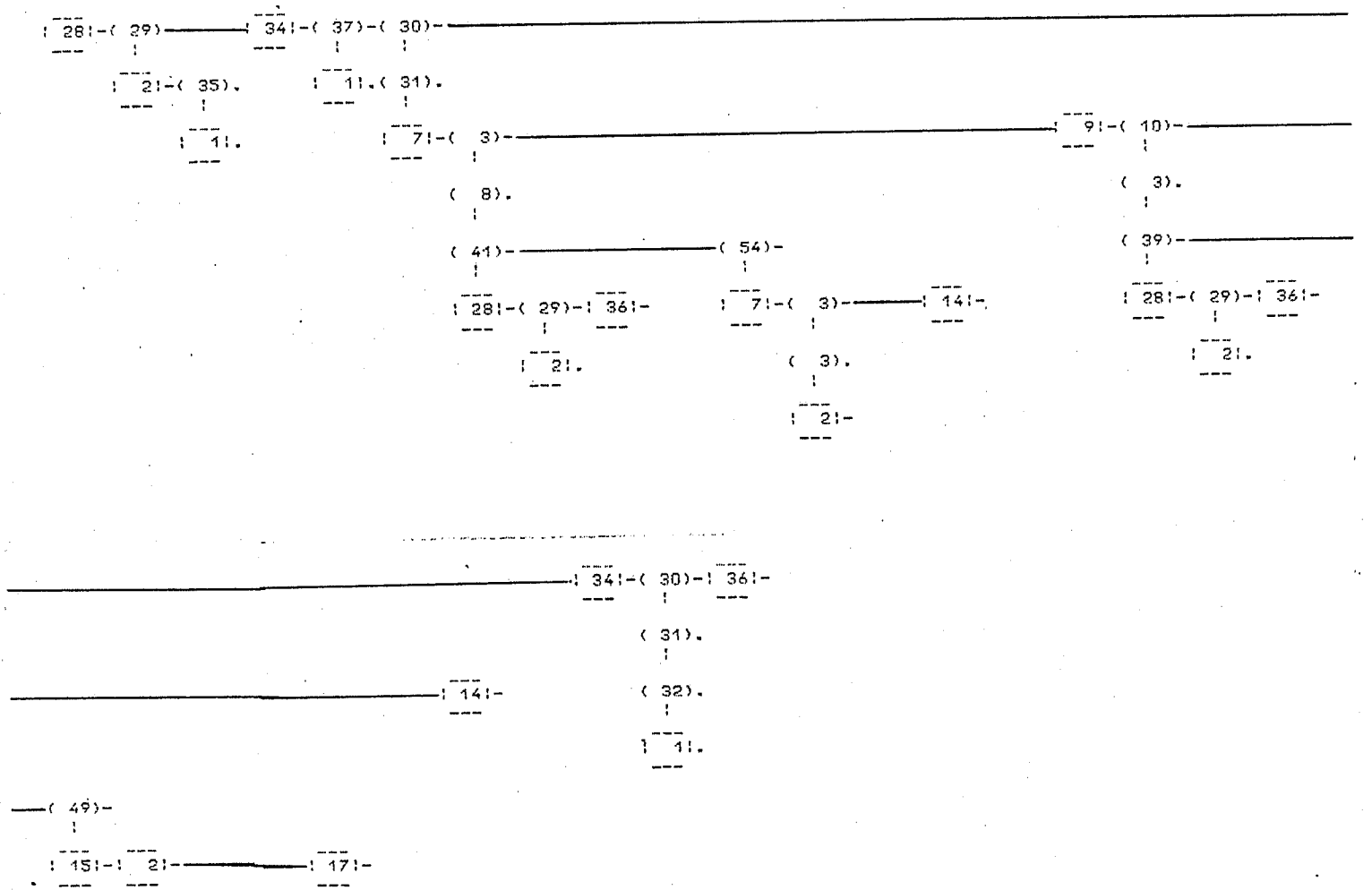
Dada uma gramática G, o algoritmo de construção da tabela de controle M do analisador sintático é o seguinte:

(1) Para cada produção A→ α da gramática, execute os passos (2) e (3).

(2) Para cada terminal a em FIRST (α), adicione A→ α na entrada M(A,a).

Fig.VII.2

Saída do Programa que Desenha Arvores



UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
NUCLEO DE COMPUTAÇÃO ELETRÔNICA

MODULO CODIFICADOR DE EXPRESSOES REGULARES — FOLHA : 1

LN	PR	IMAGEM DO CARTAO	NAO TERMINAIS NOME	COD.	TERMINAIS NOME	COD.
1	1	TRABALHO = ('EXECUTAR'/'COMPILAR') 'PARA' 'USUARIO' 'ID'	TRABALHO	200	EXECUTAR	1
					COMPILAR	2
					PARA	3
					USUARIO	4
					ID	5
2	1	'PAR-ABRE' 'ID' 'PAR-FECHA'				
					PAR-ABRE	6
					PAR-FECHA	7
3	1	('ID'/'INSTRUCOES') ':' (COMANDO ';')* 'ENCERRAR' ;				
					INSTRUCOES	8
			COMANDO	201	:	9
					;	10
					ENCERRAR	11
4	2	COMANDO = ('CRIAR' 'ACSET' 'ID' ('A' 'PARTIR' 'DE' 'ACSET' 'ID'				
					CRIAR	12
					ACSET	13
					A	14
					PARTIR	15
					DE	16
5	2	'ATE' 'VERSAO' 'NINT')? /				
					ATE	17
					VERSAO	18
					NINT	19
6	2	'ABRIR' 'ACSET' 'ID' ('COM' 'ID' & ',')?)';'				
					ABRIR	20
					COM	21
					,	22
7	2	('ABRIR' 'VOLUME' 'DE' ('ENTRADA'/'SAIDA'/'INTERACAO')				
					VOLUME	23
					ENTRADA	24
					SAIDA	25
					INTERACAO	26
8	2	'ID' ('TECLADO'/'VIDEO'/'IMPRESSORA'/'FITA'/'DISCO'/				
					TECLADO	27
					VIDEO	28
					IMPRESSORA	29
					FITA	30
					DISCO	31
9	2	'TELEIMPRESSORA') ';')*				
					TELEIMPRESSO	32
10	2	'ESTABELECER' 'PROTECAO' 'ID' ('PARA' ('LER' /'ALTERAR')				
					ESTABELECER	33
					PROTECAO	34
					LER	35
					ALTERAR	36
11	2	'SOBRE' ('ID'/'ACTRAB') & 'U') & 'E'				
					SOBRE	37

LN	PR	IMAGEM DO CARTAO	NAO TERMINAIS NOME	COD.	TERMINAIS NOME	COD.
62	8	'PAR-ABRE' EXPRESSAO 'PAR-FECHA') & ','				
63	8	/ ('CARD'/'DESAGRU°') EXPRESSAO				
					CARD	139
					DESAGRUP	140
64	8	/ ('ESTREIT' EXPRESSAO 'DE' / 'AGRUP' EXPRESSAO 'POR')				
					ESTREIT	141
					AGRUP	142
					POR	143
65	8	('ID' ('.' 'ID')?)& ','				
66	8	/ 'RENOM' EXPRESSAO 'SUBST' ('ID' 'POR' 'ID') & ','				
					RENOM	144
					SUBST	145
67	8	/ ('JUNT'/'LIGA') EXPRESSAO 'COM' EXPRESSAO ('EXCL')?				
					JUNT	146
					LIGA	147
					EXCL	148
68	8	'PAR-ABRE' ('C' 'ID' ('.' 'ID') ? ('=' / '<' / '>' /				
69	8	'<=' / '>=' / '<>') 'C' 'ID' ('.' 'ID') ?) ?				
70	8	'PAR-FECHA'				
71	8	/ 'VAZ' / 'HORA-CORR' / 'DATA-CORR'				
					VAZ	149
					HORA-CORR	150
					DATA-CORR	151
72	8	/ 'VERBETE' VERB				
			VERB	214	VERBETE	152
73	8	/ 'PAR-ABRE' EXPRESSAO 'PAR-FECHA' ;				
74	9	TERMO-CONJUNTO = 'ID' / 'AC' CONSTANTE & ';' 'FC' -				
					AC	153
					FC	154
75	9	/ EXPRESSAO / 'AC' ('VERBETE' VERB) & ';' 'FC'				
76	9	/'ACTRAB'/'ALIG'/'AREL'				
77	9	/'COLIFENS'/'DATA'/'HORA'/'INT'/'REAL'/'NUMCAR'/'LIG'				
					COLIFENS	155
					DATA	156
					HORA	157
					INT	158
					REAL	159
					NUMCAR	160
					LIG	161
78	9	/'TUP'/'TAREL'/'TALIG'/'NOME'/'VALBOOL' ;				
					TUP	162

ESTADO : 43
SIMBOLO 10 - TRANSICAO 53
SIMBOLO 14 - TRANSICAO 54
ESTADO : 44
SIMBOLO 10 - TRANSICAO 53
SIMBOLO 21 - TRANSICAO 55
ESTADO : 45
SIMBOLO 207 - TRANSICAO 56
ESTADO : 46
SIMBOLO 207 - TRANSICAO 57
ESTADO : 47
SIMBOLO 206 - TRANSICAO 58
ESTADO : 48
SIMBOLO 7 - TRANSICAO 23
SIMBOLO 10 - TRANSICAO 59
ESTADO : 49
SIMBOLO 73 - TRANSICAO 60
ESTADO : 50
SIMBOLO 5 - TRANSICAO 23
ESTADO : 51
SIMBOLO 67 - TRANSICAO 61
ESTADO : 52
SIMBOLO 50 - TRANSICAO 23
ESTADO : 53
SIMBOLO 20 - TRANSICAO 63
SIMBOLO 33 - TRANSICAO 62
ESTADO : 54
SIMBOLO 15 - TRANSICAO 64
ESTADO : 55
SIMBOLO 5 - TRANSICAO 65
ESTADO : 56
SIMBOLO 7 - TRANSICAO 66
SIMBOLO 10 - TRANSICAO 67
ESTADO : 57
SIMBOLO 7 - TRANSICAO 23
SIMBOLO 10 - TRANSICAO 68
ESTADO : 58
SIMBOLO 6 - TRANSICAO 46
ESTADO : 59
SIMBOLO 205 - TRANSICAO 48
ESTADO : 60
SIMBOLO 74 - TRANSICAO 69
SIMBOLO 75 - TRANSICAO 69
SIMBOLO 76 - TRANSICAO 69
SIMBOLO 77 - TRANSICAO 69
SIMBOLO 78 - TRANSICAO 69
SIMBOLO 21 - TRANSICAO 52
ESTADO : 43
SIMBOLO 207 - TRANSICAO 53
ESTADO : 44
SIMBOLO 207 - TRANSICAO 54
ESTADO : 45
SIMBOLO 206 - TRANSICAO 55
ESTADO : 46
SIMBOLO 7 - TRANSICAO 23
SIMBOLO 10 - TRANSICAO 56
ESTADO : 47
SIMBOLO 73 - TRANSICAO 57
ESTADO : 48
SIMBOLO 67 - TRANSICAO 58
ESTADO : 49
SIMBOLO 50 - TRANSICAO 23
ESTADO : 50
SIMBOLO 20 - TRANSICAO 60
SIMBOLO 33 - TRANSICAO 59
ESTADO : 51
SIMBOLO 15 - TRANSICAO 61
ESTADO : 52
SIMBOLO 5 - TRANSICAO 62
ESTADO : 53
SIMBOLO 7 - TRANSICAO 63
SIMBOLO 10 - TRANSICAO 43
ESTADO : 54
SIMBOLO 7 - TRANSICAO 23
SIMBOLO 10 - TRANSICAO 44
ESTADO : 55
SIMBOLO 6 - TRANSICAO 44
ESTADO : 56
SIMBOLO 205 - TRANSICAO 46
ESTADO : 57
SIMBOLO 74 - TRANSICAO 64
SIMBOLO 75 - TRANSICAO 64
SIMBOLO 76 - TRANSICAO 64
SIMBOLO 77 - TRANSICAO 64
SIMBOLO 78 - TRANSICAO 64
SIMBOLO 79 - TRANSICAO 64
SIMBOLO 80 - TRANSICAO 64
ESTADO : 58
SIMBOLO 68 - TRANSICAO 65
SIMBOLO 204 - TRANSICAO 23
ESTADO : 59
SIMBOLO 34 - TRANSICAO 66
ESTADO : 60
SIMBOLO 23 - TRANSICAO 67

```
   SIMBOLO    9 - TRANSICAO 101
ESTADO : 99F
   SIMBOLO   39 - TRANSICAO 103
   SIMBOLO   40 - TRANSICAO 102
ESTADO : 100
   SIMBOLO   10 - TRANSICAO  53
ESTADO : 101
   SIMBOLO   81 - TRANSICAO 105
   SIMBOLO   83 - TRANSICAO 104
ESTADO : 102
   SIMBOLO    3 - TRANSICAO 106
ESTADO : 103
   SIMBOLO    5 - TRANSICAO  99
   SIMBOLO   38 - TRANSICAO  99
ESTADO : 104
   SIMBOLO   84 - TRANSICAO 107
   SIMBOLO   85 - TRANSICAO 107
   SIMBOLO   86 - TRANSICAO 107
ESTADO : 105
   SIMBOLO   82 - TRANSICAO  89
ESTADO : 106
   SIMBOLO   35 - TRANSICAO 108
   SIMBOLO   36 - TRANSICAO 108
ESTADO : 107
   SIMBOLO   82 - TRANSICAO  89
   SIMBOLO   87 - TRANSICAO  89
ESTADO : 108
   SIMBOLO   37 - TRANSICAO 109
ESTADO : 109
   SIMBOLO    5 - TRANSICAO 110
   SIMBOLO   38 - TRANSICAO 110
ESTADO : 110F
   SIMBOLO   39 - TRANSICAO 111
   SIMBOLO   40 - TRANSICAO 102
ESTADO : 111
   SIMBOLO    5 - TRANSICAO 110
   SIMBOLO   38 - TRANSICAO 110
ESTADOS EQUIVALENTES --> 28 29 34 50
ESTADOS EQUIVALENTES --> 45 67
ESTADOS EQUIVALENTES --> 46 68 81 93
ESTADOS EQUIVALENTES --> 55 74
ESTADOS EQUIVALENTES --> 57 87
ESTADOS EQUIVALENTES --> 58 75
ESTADOS EQUIVALENTES --> 60 94
ESTADOS EQUIVALENTES --> 69 98
ESTADOS EQUIVALENTES --> 76101
ESTADOS EQUIVALENTES --> 78102
ESTADOS EQUIVALENTES --> 82104
ESTADOS EQUIVALENTES --> 83105
ESTADOS EQUIVALENTES --> 84106
ESTADOS EQUIVALENTES --> 88107
ESTADOS EQUIVALENTES --> 90108
ESTADOS EQUIVALENTES --> 95103109111
ESTADOS EQUIVALENTES --> 96100
ESTADOS EQUIVALENTES --> 99110
```