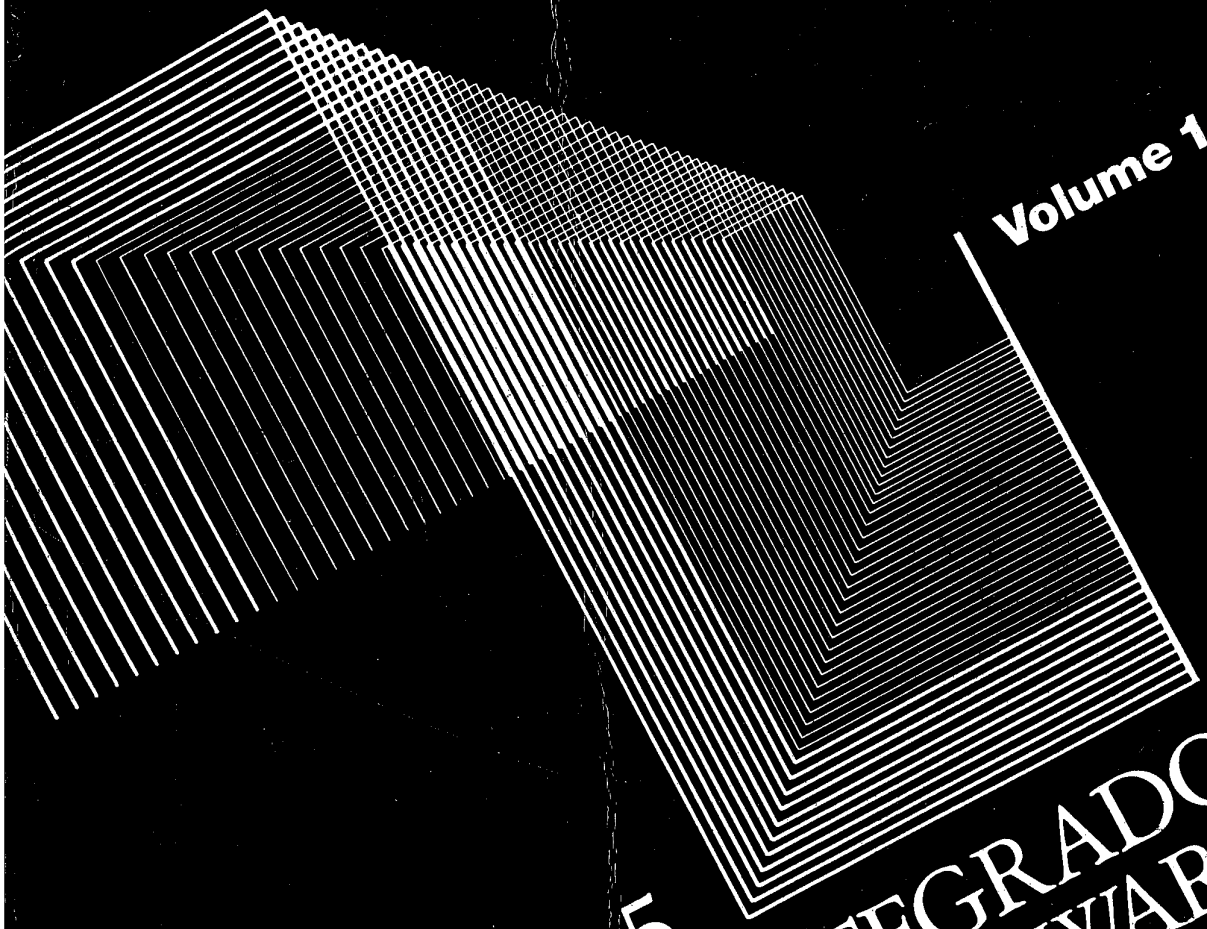

Volume 1

5 SEMINÁRIO INTEGRADO SOFTWARE E HARDWARE NACIONAIS

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Núcleo de Computação Eletrônica
CAPRE/FINEP

1978

GALÉ - Gerador de Filtros Léxicos

Conferencista e Autor:
Arndt von Staa
PUC/RJ, Informática

Resumo:

No processo de compilação a primeira atividade a ser exercida é a de decompor o texto fonte em unidades léxicas. Durante esta decomposição, serão encontradas unidades léxicas que não possuem utilidade do ponto de vista do processamento posterior, tais como comentários e sequências de "brancos". Estas unidades "inúteis" devem ser eliminadas pelo filtro léxico.

Por sua vez, as unidades léxicas úteis poderão ser símbolos representados por meio de cadeias de caracteres de tamanho variável, por exemplo identificadores de variáveis. O processamento de cadeias de tamanho variável é custoso. É desejável, então, utilizar-se uma identificação de tamanho padronizado e curto, em correspondência um para um com este símbolo. Esta compressão é efetuada por uma tabela de símbolos.

O filtro léxico é, portanto, uma rotina capaz de decompor o texto fonte em unidades léxicas, eliminar aquelas que forem inúteis, comprimir as que forem úteis através de uma tabela de símbolos, retornando identificações padronizadas, de tamanho curto e constante.

O gerador de filtros léxicos é um processador de linguagem, que traduz uma especificação de um filtro léxico dada em linguagem própria para um programa capaz de perfazer a filtragem léxica tal como especificada.

O presente trabalho apresenta os aspectos considerados mais relevantes da especificação lógica do sistema gerador de filtros léxicos. A especificação completa encontra-se em [Staa78].

1 Introdução.

No processo de compilação a primeira atividade a ser exercida é a de decompor o texto fonte em unidades léxicas. Esta atividade pode tanto ocorrer como fase isolada no processo de compilação, quanto como subatividade de um processamento mais complexo.

Quando da especificação de um sistema gerador, deverá ser imposto um mínimo de restrições, uma vez que é virtualmente impossível prever-se todas as aplicações possíveis. Consequentemente, necessita-se de interação com o usuário para fins de leitura, de impressão do texto fonte e, possivelmente, de outras tarefas.

Devido à necessidade de minimizar restrições, o sistema torna-se mais complexo, uma vez que passa a ter mais opções. Esta complexidade maior não deve incidir sobre filtros específicos criados para situações particulares. Por esta razão, optou-se por um sistema gerador produzindo não somente tabelas, mas também programas, pois, desta forma, é possível eliminar-se código irrelevante de filtros léxicos específicos.

Neste manual especificaremos primeiro como deverão funcionar os filtros léxicos e, depois, como definir filtros léxicos específicos.

2 Especificação do Filtro Léxico.

2.1 Definição.

Um filtro léxico é uma rotina que, a cada ativação, reconhece uma ou mais unidades léxicas, das quais a última foi definida como sendo útil para fases posteriores e todas as unidades léxicas anteriores à última foram definidas como sendo inúteis para fases posteriores e, portanto, deverão ser eliminadas durante a filtragem léxica.

A cada ativação o filtro léxico retorna uma identificação em correspondência um para um com a cadeia de caracteres que constitui a unidade léxica útil lida. Esta identificação, que chamaremos de lexema, possui tamanho padrão e é comprimida através de uma tabela de símbolos[DeRemer74].

São exemplos de unidades léxicas:

operadores, por exemplo: + - * / , .
variáveis
constantes
comentários- estes são eliminados durante a filtragem léxica e não são lexemas

Os filtros léxicos são gerados pelo sistema gerador de filtros léxicos a partir de uma linguagem de definição própria. Cada filtro léxico consta de tres componentes principais:

i- rotinas de controle e inicialização;

ii- transdutor finito reconhecedor de unidades léxicas que será chamado de analisador léxico;

iii- tabela de símbolos.

Os filtros léxicos permitem:

i- decompor o texto fonte em unidades léxicas conforme definição dada pelo usuário;

ii- descartar unidades léxicas definidas como inúteis para fases posteriores à filtragem léxica;

iii- retornar os lexemas correspondentes a unidades léxicas consideradas úteis para fases posteriores;

iv- ativar rotinas externas sempre que for assim determinado pelo analisador léxico definido pelo usuário;

v- descartar e/ou substituir e/ou incluir caracteres, ou sequências de caracteres, no interior dos símbolos correspondentes a lexemas;

vi- reconhecer palavras chave reservadas e/ou pré-definidas;

vii- perfazer o reconhecimento condicional de seções componentes de uma unidade léxica;

viii- permitir o acesso a variáveis internas através de parâmetros passados para rotinas do usuário.

Os filtros léxicos não farão:

i- impressão dos dados lidos. Isto somente poderá ser feito através de rotinas fornecidas pelo usuário;

ii- leitura do texto fonte. Isto somente poderá ser feito através de rotina fornecida pelo usuário. Esta será ativada por comando efetuado pelo analisador léxico. Este comando estará usualmente associado ao reconhecimento da condição de fim de texto na área contendo o texto a ser analisado léxicamente.

2.2 Entrada.

Os filtros léxicos não são capazes de perfazer a leitura do texto fonte. Tal operação é efetuada através de uma rotina fornecida pelo usuário. Esta rotina terá seu nome definido na seção <comunicação> da definição do filtro léxico.

Sempre que a rotina de leitura for chamada, deverá ser devolvido por ela um ponteiro (endereço) para uma área que contém o texto fonte (ou parte dele) a ser analizado léxicamente.

É seguinte o formato da área de texto a ser analisada léxicamente:

TAM	POS	c1	c2	...	cTAM

TAM- indica o tamanho da cadeia de caracteres contida na área de texto a ser analisado léxicamente ("buffer" de entrada). Este valor não é alterado pelo filtro léxico.

POS- contém o índice do próximo caracter a ser lido. Este valor deve ser inicializado sempre pela rotina de leitura fornecida pelo usuário. A inicialização deverá ser tal que POS indique para o primeiro caracter da porção da cadeia a ser analisada léxicamente. O valor inicial de POS deverá estar contido no intervalo $1 \leq POS \leq TAM$. O valor de POS é continuamente alterado pelo filtro léxico.

ci- onde $1 \leq i \leq TAM$ são os caracteres a serem lidos. Estes caracteres poderão ser alterados pelo filtro léxico.

Condições especiais:

i- se $TAM \leq 0$, é reconhecido o caracter especial $FARQ - fim de arquivo.

ii- se $POS > TAM$, é reconhecido o caracter especial $FBUF - fim de "buffer". Ao ser encontrado este caracter, a rotina de leitura fornecida pelo usuário será acionada, desde que esta operação esteja associada ao reconhecimento do caracter especial $FBUF.

iii- se a rotina de leitura fornecida pelo usuário retornar o ponteiro nulo, será reconhecido o caracter $FIM. Uma vez tendo sido reconhecido $FIM, o filtro léxico passará a reconhecer somente este caracater especial. Isto ocorrerá independente de chamadas de rotinas fornecidas pelo usuário e associadas ao reconhecimento de $FIM. O reconhecimento do caracter $FIM causa, também, a transferência para arquivo do conteúdo da tabela de símbolos neste momento residente na memória principal.

iv- diversas condições poderão levar o filtro léxico a reconhecer o caracter especial $ERRO. São seguintes as condições:

a- o transdutor léxico é incapaz de reconhecer pelo menos uma unidade léxica, útil ou não;

b- o transdutor léxico atigiu um estado definido como inconsistente pelo usuário. Associado a este estado existe uma mensagem de erro criada pelo usuário;

c- o transdutor léxico atingiu um estado definido como inconsistente pelo próprio sistema gerador de filtros léxicos. Associado a este estado existe uma mensagem de erro criada pelo sistema. Exemplos de inconsistências deste gênero são: esgotamento da capacidade de armazenagem de símbolos, tamanho de símbolo excessivamente grande etc.

Em todos estes casos de reconhecimento de condição de erro, o reconhecimento em curso será suspenso e será gerado um lexema indicativo de erro. Em alguns casos, será possível recomeçar a operação do filtro léxico imediatamente após ao ponto em que foi encontrada a condição de erro.

v- se e somente se a expressão condicional contida em um reconhecimento condicional for verdadeira, é reconhecido o caracter especial $COND. Caso esta condição seja falsa, nada será reconhecido por esta expressão. A qualquer instante pode ser reconhecido o "caracter" vazio. Isto corresponde ao reconhecimento de $COND onde a condição de reconhecimento é sempre verdadeira.

Cabe ao usuário evitar tentativas de leitura além do final de arquivos de texto fonte, através da especificação adequada dos filtros léxicos e/ou através do controle de ativacões da rotina de leitura fornecida.

A sistemática de leitura será tal que o usuário do gerador do filtro léxico tenha a impressão de estar lendo uma cadeia "infinita" de caracteres do conjunto:

$$\text{ALFABETO} \cup \{\$FARQ, \$FBUF, \$FIM, \$ERRO, \$COND\}$$

A rotina de leitura fornecida pelo usuário deverá ser capaz de selecionar arquivos e de salvaguardar registros ainda não completamente analisados caso seja permitida a leitura de texto fonte a partir de diversos arquivos e/ou áreas contendo texto fonte.

Os caracteres especiais $FARQ, $FBUF, $FIM, $ERRO e $COND não possuem representação binária própria. Sendo assim, também não podem fazer parte da cadeia de caracteres que constitue um símbolo na tabela de símbolos. Estes caracters cor-

respondem a condições lógicas reconhecidas pelo filtro léxico. Estas condições muitas vezes não são repetitíveis, uma vez que dependem diretamente das condições de contorno no momento em que foram reconhecidas.

2.3 Resultados Produzidos.

Para cada ativação do filtro léxico será retornado um descritor de lexema, possuindo o seguinte formato:

POS	TIPO	ID

POS- é o índice do primeiro caracter na primeira área de entrada, utilizado para o reconhecimento da unidade léxica. Este índice independe da possibilidade de descarte de caracteres que constituem a unidade léxica lida, não sendo, portanto, transcrito para o símbolo correspondente ao lexema. Por exemplo, quando for reconhecida uma constante "cadeia de caracteres" obedecendo as convenções do PL/1, ambas as aspas inicial e final devem ser descartadas do símbolo antes de pô-lo na tabela de símbolos, o índice POS indica porém, o local onde estava a aspa inicial da cadeia. O índice POS é particularmente útil para a geração de mensagens de erro vinculadas ao texto fonte, onde estas mensagens são produzidas em fases posteriores à análise léxica.

TIPO- é um valor numérico que determina a classe do lexema encontrado. Este valor é atribuido pelo transdutor finito reconhecedor. Os valores possíveis para TIPO são determinados durante a definição do filtro léxico. O TIPO será utilizado em fases posteriores à análise léxica para distinguir lexemas diferentes possuindo símbolos iguais. Além disso, permite tomar-se decisões durante a análise sintática sem que se tenha que recorrer ao símbolo contido na tabela de símbolos.

ID- é um valor numérico em correspondência um para um com o símbolo correspondente ao lexema reconhecido. O valor de ID é determinado pela tabela de símbolos.

Além desses resultados produzidos a cada ativação do filtro léxico efetuada pelo usuário, é produzido ainda um arquivo contendo a tabela de símbolos. Desta forma, poder-se-á ter acesso aos símbolos contidos na tabela, através de chamadas a rotinas específicas, tanto durante a filtragem léxica, quanto em fases posteriores.

Lexemas de erro têm sempre TIPO=0 e ID é a identificação de uma mensagem de erro explicando o erro havido. Os lexemas de erro são gerados imediatamente ao encontrar-se a condição de erro. Em muitos casos a condição de erro é recuperável em outros será final. No caso de uma condição de erro recuperável, a unidade léxica sendo reconhecida poderá continuar a ser reconhecida. Desta forma é possível continuar-se com a análise léxica, sintática e semântica, sem que o erro detectado provoque a geração de outros a nível sintático ou semântico. É claro que a condição de erro a nível léxico não é corrigida, no sentido estrito da palavra, porém é "remendada" o suficiente para que se possa prosseguir sem uma geração de erros consequência do erro encontrado.

Através da consulta ao campo ID e/ou aos caracteres iniciais da mensagem de erro, o usuário poderá determinar se se trata de um erro recuperável ou final. Em se tratando de um erro recuperável, o usuário poderá passar controle ao ponto de entrada de recuperação de erro ($RECUPERE_ERRO()). Desta forma o reconhecimento da unidade léxica onde foi encontrado o erro é continuado. Caso o filtro léxico seja ativado através de seu ponto de antrada normal ($FILTRO_LEXICO()), é iniciado o reconhecimento de uma nova unidade léxica, sendo perdida aquela em que ocorreu o erro. Caso o usuário ative o filtro léxico através do ponto de entrada de recuperação de erro quando o último lexema encontrado na instância ativa não havia sido lexema de erro recuperável, será gerado um lexema de erro final. Isto pode ocorrer se o último lexema encontrado nesta instância do filtro léxico tenha sido um lexema normal ou de erro final. Cabe frisar que é de inteira responsabilidade do usuário identificar se o lexema retornado é normal, de erro recuperável, ou de erro final. É seguinte a convenção adotada pelo gerador de filtros léxicos

a- erros recuperáveis têm como primeiro caracter da mensagem de erro um caracter diferente de asterisco (*).

b- erros finais têm como primeiro caracter da mensagem de erro o caracter asterisco (*).

As mensagens de erro poderão ser geradas devido a inconsistências reconhecidas pelo usuário ou pelo próprio filtro léxico. As mensagens de erro do usuário são especificadas durante a definição do filtro léxico ($MENSAGEM(...) e $ABORTE(...)). As mensagens de erro do sistema serão descritas em outra publicação (Manual de Especificação Física) sendo redigida no momento.

Além dos lexemas, é produzido, também, um arquivo contendo a tabela de símbolos. Este arquivo é de acesso direto, sendo que seu formato será descrito em outra publicação (Manual de Especificação Física) sendo redigida no momento. Ao encontrar o caracter especial $FIM, o arquivo da tabela de símbolos é atualizado automaticamente, garantindo-se que esteja completo após esta atualização.

Em fases de processamento posteriores à filtragem léxica, este arquivo poderá ser acessado através de um conjunto de rotinas de acesso à tabela de símbolos. Este conjunto também é gerado pelo gerador de filtros léxicos.

2.4 Ativação.

O filtro léxico é ativado através de chamada a um de seus pontos de entrada. Os nomes destes pontos de entrada são definidos na seção <comunicação> da especificação do filtro léxico.

O ponto de entrada normal ($FILTRO_LEXICO()) sempre inicia o reconhecimento de um novo lexema, mesmo que o último lexema retornado pela instância ativa do filtro léxico tenha sido um lexema de erro recuperável.

O ponto de entrada de recuperação de erro ($RECUPERE_ERRO()) permite continuar no reconhecimento da unidade léxica em que foi encontrado um erro recuperável. Caso o último lexema retornado pela instância ativa do filtro léxico não seja lexema de erro recuperável, será gerado um lexema de erro final.

O retorno ocorrerá somente após o reconhecimento de um lexema. Como lexemas correspondem somente a unidades léxicas úteis, cabe ao usuário definir o filtro léxico de tal forma que não sejam induzidos "loops" infinitos de reconhecimento de unidades sintáticas inúteis. Atenção especial deve ser dada aos caracteres especiais $FARQ, $FBUF, $FIM e $COND, uma vez que estes podem facilmente induzir "loops" infinitos por poderem ser reconhecidos em sequências infinitas.

O reconhecimento do caracter $ERRO corresponde, sempre, a um lexema de erro. O usuário deverá tomar o cuidado de eliminar a causa do erro antes de reativar o filtro léxico, pois, de outra forma, poderá ser induzido um ciclo infinito de reconhecimento de condições de erro.

3 Especificação das Rotinas Componentes do Filtro Léxico.

Por razões de espaço disponível para este texto, apresentamos aqui somente um resumo da especificação do filtro léxico. Uma especificação competa encontra-se em [Staa78].

Cada filtro léxico específico é composto pelas rotinas:

i- analisador léxico;

ii- tabela de símbolos;

iii- rotina de inicialização.

O analisador léxico tem por objetivos:

i- decompor o texto fonte em unidades léxicas;

ii- comunicar as porções úteis das cadeias de caracteres que perfazem unidades léxicas úteis para a área de comunicação da tabela de símbolos;

iii- efetuar a comunicação com o exterior através da ativação de rotinas fornecidas pelo usuário, bem como através da ativação do próprio filtro léxico comandada pelo usuário.

A tabela de símbolos tem por objetivos:

i- determinar um valor em correspondência um para um com os símbolos correspondentes a unidades léxicas úteis (lexemas);

ii- permitir a salvaguarda dos símbolos contidos na tabela para acesso tanto durante a filtragem léxica, quanto durante fases posteriores;

iii- prover rotinas de acesso a símbolos a partir de sua identificação.

A rotina de inicialização tem por objetivo criar instâncias de filtros léxicos e inicializar todas as variáveis a serem mantidas por uma determinada instância.

A análise léxica é a decomposição do fluxo de entrada em unidades léxicas.

A análise léxica é efetuada por um transdutor finito [Gries 1971, Hopcroft 1969, Aho 1972]. É função do transdutor finito:

i- o reconhecimento de unidades léxicas, úteis ou não;

ii- a eliminação de unidades léxicas inúteis;

iii- a substituição de caracteres de partes do símbolo correspondentes a unidades léxicas úteis;

iv- o reconhecimento de palavras reservadas e/ou pré-definidas;

v- a ativação de rotinas fornecidas pelo usuário sempre que for efetuada uma transição requerendo esta ativação;

vi- a intercomunicação com a tabela de símbolos;

vii- a geração de lexemas indicativos de erro.

Símbolos correspondem à transdução de cadeias de caracteres das unidades léxicas encontradas. Um símbolo poderá ser qualquer cadeia de zero ou mais caracteres, sendo o seu tamanho virtualmente ilimitado. É função da tabela de símbolos estabelecer uma correspondência um para um entre a identificação do símbolo, campo ID do lexema, e o símbolo propriamente dito. Esta identificação permite recuperar-se o símbolo sem maiores dificuldades.

A tabela de símbolos é constituida por dois mapeamentos. Um mapeando cadeias de caracteres, os símbolos, sobre números naturais, e o outro mapeando números naturais sobre os símbolos correspondentes, ou, então, sobre o símbolo indefinido ϕ. Formalmente:

TS_1 : Símbolos -> N
TS_2 : N -> Símbolos $\cup \{ \phi \}$

A cada símbolo corresponde exatamente um número natural e a cada número natural corresponde exatamente um símbolo ou, então, o símbolo indefinido ϕ, tal que:

$TS_2(TS_1(S)) = S$
se $TS_2(n) \neq \phi$ então $TS_1(TS_2(n)) = n$

Convém enfatizar que a finalidade única e exclusiva da tabela de símbolos é a de converter cadeias de caracteres em números naturais e vice-versa. Não é função da tabela de símbolos manter indicações quanto aos atributos (tipos) dos símbolos nela contidos. A tabela de atributos deve ser criada externamente ao filtro léxico[DeRemer74]. Tampouco é função do filtro léxico converter constantes numéricas em suas representações binárias internas. Caso isto seja desejado, o usuário deverá prover uma rotina para perfazer esta conversão.

O mapeamento TS_1 é chamado de instalação de símbolo, e é composto pelas seguintes operações:

a- pesquisar a tabela de símbolos à procura do símbolo s a instalar;

b- caso o símbolo a instalar já exista, devolver a identificação com que existe na tabela de símbolos;

c- caso o símbolo a instalar ainda não exista, gerar uma identificação para ele, incluir a identificação e o símbolo na tabela de símbolos, e devolver a identificação criada.

O mapeamento TS_2 é chamado de recuperação de símbolo e é composto pelas seguintes operações:

a- verificar se a identificação fornecida corresponde a um símbolo existente;

b- caso corresponda, devolver o símbolo correspondente;

c- caso não corresponda, devolver o símbolo indefinido.

4 Linguagem de Especificação de Filtros Léxicos.

Por limitações de espaço para este texto, apresentamos aqui somente uma fração da especifição da linguagem de especificação de filtros léxicos. A porção escolhida é bastante representativa do potencial desta linguagem. No apêndice é apresentada uma especificação completa de um filtro léxico escrito na linguagem de especificação. A descrição completa da linguagem de especificação de filtros léxicos encontra-se em [Staa78].

A linguagem de especificação de filtros léxicos está baseada na notação utilizada no compilador de compiladores COMCOM[Staa70]. Esta notação, por sua vez, é inspirada na notação de Kleene para linguagens regulares[Hopcroft69] e na notação Backus Naur Form (BNF) [Naur63].

Para a compreensão deste capítulo é assumido o leitor ter conhecimento de linguagens regulares, bem como das notações mencionadas. Detalhes quanto a estas notações poderão ser encontrados em [Hopcroft69, Gries71, Aho72, Bauer74].

A linguagem de especificação de filtros léxicos descreve de uma só vez tanto a linguagem regular reconhecida pelo transdutor (o conjunto de unidades léxicas permitidas), quanto as operações de transdução a serem efetuadas por este transdutor para produzir símbolos.

A linguagem de especificação de filtros léxicos é definida em termos de operações de reconhecimento, operações de transdução e de ações a serem efetuadas caso o reconhecimento chegue a um determinado ponto. O princípio adotado é semelhante ao adotado em SNOBOL4[Griswold71].

A leitura de áreas de texto de entrada, usualmente linhas, efetuada através de rotina fornecida pelo usuário, simula a existência de um fluxo de entrada contínuo e ininterrupto. Este efeito é conseguido através de chamadas à rotina de leitura do usuário sempre que uma linha de texto de entrada exaurir. O fluxo ininterrupto é conseguido através da concatenação das linhas de texto de entrada simulando a existência entre elas do caracter especial $FBUF. Sobre este fluxo de entrada é movimentado um cursor. O reconhecimento de uma cadeia é efetuado a partir do caracter sob este cursor. O reconhecimento somente sucederá, caso o fluxo de entrada contenha caracteres exatamente iguais aos especificados pela cadeia a reconhecer. As cadeias poderão ser especificadas como sendo cadeias constantes, valores do tipo $CADEIA contidos em variáveis, valores do tipo $CADEIA retornados por funções do usuário ou do sistema, ou, finalmente, um dos elementos de um conjunto de cadeias.

Dada a condição de linguagem regular, o reconhecimento poderá ser efetuado de maneira determinística, exceto quando a cadeia a reconhecer for resultado da avaliação de funções ou conteúdos de variáveis. Estas cadeias, porém, não poderão conter os caracteres especiais $FIM, $FBUF, $FARQ, $COND ou $ERRO. Por esta razão, caso uma cadeia seja reconhecível ela estará integralmente contida no registro de entrada. Consequentemente, mesmo no caso de reconhecimento de cadeias variáveis, este reconhecimento poderá ocorrer deterministicamente.

Cada especificação de filtro léxico consta das seguintes quatro porções:

i- cabeçalho - onde são dadas informações sobre o título, autor e datas, e descrição do filtro léxico;

ii- comunicação - onde é definida a relação do filtro léxico a ser gerado com outros programas. A seção de comunicação descreve os pontos de entrada do filtro léxico gerado inúteis para fases posteriores, as referências a rotinas a serem providas pelo usuário, os nomes de áreas externas e os nomes das seções de controle criadas pelo gerador de filtros léxicos;

iii- definição - onde são definidos os lexemas, os nomes de unidades sintáticas, conjuntos de cadeias de caracteres e variáveis a serem utilizadas pelo filtro léxico;

iv- transdutor - onde são definidas as unidades léxicas reconhecíveis, bem como as transduções a serem efetuadas.

É seguinte a sintaxe para a definição de filtros léxicos:

```
<filtro léxico> ::= <cabeçalho>
                    <comunicação>
                    <definição>
                    <transdutor>
                    ($FIM | "$OUTRO" <filtro léxico>).
```

Poderão ser definidos vários filtros léxicos em uma somente aplicação. Estas definições deverão figurar em sequência, separadas pala palavra $OUTRO. O término da última especificação de filtro léxico é sinalizada pelo caracter especial $FIM (ponteiro nulo retornado pela rotina de leitura).

4.2 Unidades Léxicas da Linguagem de Especificação.

São seguintes os elementos léxicos da linguagem de especificação de filtros léxicos:

i- cadeias - sequências de caracteres. Cadeias descrevem itens a reconhecer, bem como incorporações aos símbolos a serem efetuadas por transdução;

ii- números - sequências de dígitos. Números são sempre maiores ou iguais a zero;

iii- não terminais - elementos gramaticais que descrevem conjuntos de subsequências válidas que poderão ocorrer no interior de unidades léxicas;

iv- nomes - palavras reservadas ou palavras definidas pelo usuário, possuindo significado especial próprio;

v- operadores e sinais de pontuação;

vi- comentários e sequências de um ou mais caracteres em branco - texto fornecido pelo usuário para tornar a especificação mais legível e compreensível para o leitor. Comentários e sequências de caracteres em branco são ignorados pelo gerador de filtro léxicos, servindo somente como separadores de elementos léxicos. O uso de separador é mandatório somente se a sua ausência puder causar ambiguidades.

Descreveremos a seguir os elementos que constituem a linguagem de especificação de filtros léxicos. A descrição utiliza a própria linguagem de especificação para definir os elementos léxicos.

O texto a seguir descreve os lexemas utilizados pelo gerador de filtros léxicos. Estes lexemas estão descritos na própria linguagem de especificação de filtros léxicos. Desta forma tem-se uma definição precisa, se bem que um pouco difícil de compreender em alguns casos, pois necessita-se saber o que se está tentando apreender...

No gerador de filtros léxicos as áreas de texto de entrada correspondem a linhas de texto fonte. Chamaremos, então, de linha as áreas de texto fonte.

4.2.1 <Cadeia>.

Cadeias definem sequências de zero ou mais caracteres que poderão ser utilizadas para reconhecimento ou para a incorporação aos símbolos através de transdução. O início das cadeias de caracteres é sinalizado pelo caracter aspa dupla ("). O final da cadeia de caracteres é sinalizado por outro caracter aspa dupla não precedido pelo caracter asterisco (*) ou por fim de linha de texto de entrada ($FBUF). Cadeias são definidas da seguinte forma:

```
<cadeia> ::=
   ( <cadeia normal> |
     <caracter especial reconhecível> ).

<caracter especial reconhecível> ::=
     <nome> $NO_CONJUNTO ("$FIM" | "$FARQ" | "$FBUF" ).

<cadeia normal> ::= $ELIMINE "*""  <corpo da cadeia>
     $ELIMINE <fim da cadeia> .

<fim da cadeia> ::=
   ( "*"" |
     $FBUF $LE_LINHA() ).

<corpo da cadeia> ::= 0-(
   ( ¬("**" | "*"") |
     $ELIMINE "**" <operação de cadeia> ) ).

/* <corpo da cadeia> termina somente se for encontrado
/* aspa dupla (") ou um caracter especial que é $FBUF

<operação de cadeia> ::=
   ( "*"" |
     "**" |
     $TRADUZA $HEXA_BYTE(<byte hexadecimal>) |
     $ELIMINE <concatenador> |
     $MENSAGEM("0001Erro de operação em cadeia.") ).

<byte hexadecimal> ::= 2 ($DIGITO_HEXADECIMAL).
```

```
<concatenador> ::= "-" $LE_LINHA() <concatenação>.

<concatenação> ::= 0-(" ")
   ( "*"" |
     $MENSAGEM("0002Erro de concatenação de cadeia.")
                  $ENCONTROU <cadeia> ).
```

Durante o reconhecimento de uma cadeia, poderão ser encontrados comandos que atuam sobre esse reconhecimento. Através destes comandos torna-se possível a introdução de caracteres que não figurem no teclado do dispositivo de entrada utilizado para gerar a especificação; a introdução de caracteres com significado especial; e a geração de cadeias maiores do que as limitações impostas pelo equipamento de entrada utilizado.

Todos os comandos que atuam sobre o reconhecimento de cadeias são sinalizados pelo caracter de escape asterisco (*). São seguintes os comandos permitidos:

*" introdução do caracter aspa dupla no corpo da cadeia;

** introdução do caracter asterisco no corpo da cadeia;

*HH introdução no corpo da cadeia do caracter cuja representação em hexadecimal é HH;

*- concatenação de linhas de texto de entrada. A linha corrente é terminada imediatamente ao encontrar-se este par de caracteres. A leitura da cadeia é reiniciada imediatamente após ter sido encontrado um caracter aspa dupla na linha de texto de entrada logo a seguir. Caso este caracter não seja encontrado, ou não seja o primeiro caracter não branco encontrado, é emitida uma mensagem de erro recuperável. Após a mensagem de erro é terminada a leitura da <cadeia>.

4.2.2 <Não Terminal>.

<Não terminal> é uma sequência de letras, dígitos, caracteres em branco e caracteres do conjunto :$, @, #, _: compreendidos entre os parênteses angulares < e >. O primeiro caracter do <não terminal> não deve ser dígito. Letras minúsculas serão traduzidas automáticamente para letras maiúsculas. Sequências de caracteres em branco serão traduzidas para exatamente um caracter em branco, exceto sequências logo após ao abre perêntese angular (<) e imediatamente antes do fecha perêntese angular (>) que serão eliminadas. O <não terminal> deve estar todo compreendido em uma linha de texto fonte.

```
"ABC"                 a cadeia de corpo ABC
""                    a cadeia com corpo vazio
"*""                  a cadeia de corpo "
"**"                  a cadeia de corpo *
"*F1a***F2b*"*F3"     a cadeia de corpo 1a*2b"3

"AB*-
      "CD*-           a cadeia de corpo ABCDEF
  "EF"

"ABC¥¥$FBUF           a cadeia de corpo ABC¥¥
     (Obs. ¥ caracter em branco, $FBUF fim de linha)

"AB*-
    X"CD"             a mensagem de erro 0002
```

Figura 4.1 Exemplos de <cadeia>.

É seguinte a sintaxe dos <não terminais>:

```
CONJUNTO  CARACTERES_DE_NOME = ($LETRA | $DIGITO |
      "$" | "@" | "#" | "_").

<não terminal> ::= $ELIMINE "<"
      ( $TRADUZA $MAIUSCULAS(<corpo não terminal>)
                 $ELIMINE <fim não terminal> |
        $MENSAGEM("0003Não terminal errado.") ).

<corpo não terminal> ::= $ELIMINE <brancos>
                         <resto corpo>.

<resto corpo> ::=
        CARACTERES_DE_NOME ~($DIGITO)
        0-( ( CARACTERES_DE_NOME |
              " " $ELIMINE <brancos> ) ).

<fim não terminal> ::=
      ( $SE($ULTIMO_TRANSFERIDO = " ") $RETIRE(1) |
            "" ) ">".

<brancos> ::= 0-(" ") .
```

```
<não terminal>
<   não   terminal    >
< NãO TeRmInAl >

correspondem todos ao não terminal de corpo:
            "NÃO TERMINAL"
```

Figura 4.2 Exemplos de <não terminal>.

4.2.3 <Nome>.

Um <nome> é uma sequência de letras, dígitos e caracteres do conjunto : $, @, #, _:. O primeiro caracter de um <nome> não deve ser dígito. Letras minúsculas serão convertidas automaticamente para letras maiúsculas. Não podem existir caracteres em branco nos nomes. Nomes serão utilizados para identificar variáveis, funções, operações, conjuntos e comunicadores.

Os nomes pré-definidos pelo gerador de filtros léxicos e que principiam pelo caracter $, são reservados. Nomes utilizados pelo gerador de filtros léxicos que não principiem pelo caracter $ poderão ser redefinidos pelo usuário a qualquer momento. O usuário poderá definir nomes principiando pelo caracter $, desde que não sejam iguais a nomes pré-definidos pelo gerador de filtros léxicos. Isto, porém, não é boa prática, uma vez que poderá confundir o leitor da especificação do filtro léxico.

É seguinte a sintaxe para <nomes>:

```
CONJUNTO  CARACTERES_DE_NOME = ( $LETRA | $DIGITO |
                "$" | "@" | "#" | "_").

<nome> ::= $TRADUZA $MAIUSCULAS(<corpo do nome>).

<corpo do nome> ::= CARACTERES_DE_NOME ¬($DIGITO)
                0-( CARACTERES_DE_NOME ).
```

```
Ab_C          equivale a:  AB_C
$#@_X
$
$LETRA
$letra      equivale a: $LETRA
```

Figura 4.3 Exemplos de <nomes>.

4.2.4 <Comentário> e <Sequência em Branco>.

<Comentário> é qualquer sequência de caracteres iniciada pelo par de caracteres /*, e terminada ou pelo par */ ou por fim de linha encontrado no fluxo de entrada.

<Sequência em branco> é uma sequência de um ou mais carcateres em branco e/ou fim de linha ($FBUF).

<Comentário> e <sequência em branco> poderão ocorrer em qualquer lugar entre elementos léxicos da linguagem de especificação de filtros léxicos. Seu único efeito durante a filtragem léxica de uma especificação de filtro léxico é de delimitar componentes léxicos desta especificação, de resto são ignorados pelo gerador de filtros léxicos.

Cada <comentário> está compreendido no máximo em uma linha. Por outro lado, uma <sequência em branco> poderá ocupar seções de uma linha, linhas inteiras, podendo, inclusive, continuar na próxima linha.

É seguinte a sintaxe de <comentário> e <sequência em branco>:

```
<comentário> ::= "/**" 0-( ( ¬("**") | "**" ¬("/") )
                 ( "**" | "" ) ("/" | $FBUF $LE_LINHA() ) .

<sequência em branco> ::= 1-((" " | $FBUF $LE_LINHA() )).
```

4.2.5 <Número> e Operadores.

Números são sequências de um ou mais dígitos decimais. Números são sempre maiores ou iguais a zero. É seguinte a sintaxe de <número>:

```
<número> ::= 1-($DIGITO).
```

O conjunto de operadores será definido à medida que forem apresentadas as construções permitidas na linguagem de especificação de filtros léxicos. No apêndice A é apresentada a especificação completa do filtro léxico utilizado pelo gerador de filtros léxicos. Neste apêndice são apresentados, também, os operadores existentes na linguagem de especificação de filtros léxicos.

5 Epílogo.

Uma especificação dificilmente se presta para uma seção de conclusões, em particular se a especificação é truncada tal como a presente. Apresentaremos então alguns argumentos justificando o esforço desprendido no projeto e implementação.

Na PUC/RJ são diversas as linguagens sendo implementadas e/ou sendo projetadas para implementação no futuro próximo. Só isto já bastaria para justificar o desenvolvimento de uma ferramenta de suporte à implementação de processadores de linguagem.

Sabe-se hoje, também, que em muitos casos sistemas são fracassos devido a uma modalidade de comunicação homem-máquina inadequada. O gerador de filtros léxicos poderá servir, então, para o suporte de um laboratório experimental onde desenvolver novas técnicas de comunicação homem-máquina. Projetos dessa natureza também estão em andamento na PUC/RJ.

Referências Bibliográficas.

Aho, A.V.; Ullman, J.D.
The Theory of Parsing, Translation and Compiling; Volume 1: Parsing, 1972; Volume 2: Compiling, 1973; Prentice Hall

Bauer, F.L.; Eickel, J. editores
Compiler Construction, An Advaced Course; Springer Verlag, Lecture Notes in Computer Science no. 21, Berlin; 1974

Brown, M.P.
Estudo Comparativo de Tabelas de Símbolos; Tese de Mestrado, Departamento de Informática, Pontifícia Universidade Católica, Rio de Janeiro; 1971

DeRemer, F.L.
'Lexical Analysis'; em [Bauer74]; 1974; pags 109-120

Gries, D.
Compiler Construction for Digital Computers; John Wiley and Sons, New York; 1971

Griswold, R.E.; Poage, J.F.; Polonsky, I.P.
The SNOBOL4 Programming Language; Prentice Hall, Englewood Cliffs, New Jersey; Segunda Edição, 1971

Hopcroft, J.E.; Ullman, J.D.
Formal Languages and Their Relation to Automata; Addison Wesley Publishing, Co., Reading, Massachusetts; 1969

Johnson, W.R.; Porter, J.S.; Ackley, S.I.; Ross, D.T.
'Automatic Generation of Efficient Lexical Processors Using Finite State Techniques'; em Communications of the Association for Computing Machinery 11(12), New York; 1968; pags 805-813

Knuth, D.E.
The Art of Computer Programming, Volume 3: Sorting and Searching; Addison Wesley Publishing Co., Reading, Massachusetts, 1973

Naur, P. editor
'Revised Report on the Algorithmic Language ALGOL60'; em Numerische Mathematik 4; Springer Verlag, Berlin; 1963; pags 420-453

Staa, A. v.
COMCOM - Compilador de Compiladores; Grupo de Aplicacões, Rio Datacentro, Pontifícia Universidade Católica, Rio de Janeiro, 1970.

Staa, A. v.
Especificação do Projeto COMCOM2; Monografias em Ciência da Computação no. 16/77, Departamento de Informática, Pontifícia UniversidadeCatólica, Rio de Janeiro; Outubro 1977.

Staa, A. v.
Gerador de Filtros Léxicos, Manual de Especificação Lógica; Monografias em Ciência da Computação, Departamento de Informática, Pontifícia Universidade Católica, Rio de Janeiro; 1978