SCC0605 - Teoria da Computação e Compiladores
Docente: Prof. Thiago A. S. Pardo

Trabalho 1: Analisador Léxico para Linguagem P--

Alunos:
Bruno Ribeiro Helou (10276852)
Juan Carlos Elias Obando Valdivia (7487156)
Maria Luisa do Nascimento da Silva (10310721)

São Carlos
Junho 2021

Sumário

1. Decisões de Projeto

Para a implementação do analisador léxico, foi escolhida a linguagem de programação Python 3, pelo fato dos integrantes do grupo já terem maior familiaridade com a linguagem, mesmo não sendo a escolha de maior velocidade de execução. Assim como, por ser uma linguagem de mais alto nível do que outras opções, Python foi considerada mais adequada para os fins didáticos do trabalho.

Para a implementação do analisador, os comentários serão reconhecidos e ignorados, mas quando o comentário não é fechado (falta do símbolo "}"), é indicado erro léxico de comentário não finalizado. Já os caracteres não imprimíveis (espaço, tabulação e quebra de linha) são retirados durante a análise de cada cadeia e são considerados possíveis finais para números, identificadores e palavras reservadas.

O autômato finito construído para a análise léxica reconhece as cadeias mapeadas na Tabela 1, assim como as palavras reservadas da linguagem - como por exemplo "for" e "to" que são adicionadas para a inclusão do comando for -, identificadores, números reais e inteiros. O autômato retorna as cadeias de caracteres reconhecidas juntamente com seu token (mapeado na seção 2.a), além de indicar erros de caractere inválido, identificador e número mal formado, com o intuito de reportar mensagens de erros intuitivas que auxiliem o usuário e entender e consertar os erros encontrados.

O código foi estruturado da seguinte maneira: o arquivo *main.py* recebe um arquivo de entrada com o código a ser analisado e chama a função *make_lexical_analysis* do arquivo *lexical_analysis.py* que quebra a entrada em cadeias e realiza a análise léxica utilizando as funções auxiliares definidas neste mesmo *script* para o reconhecimento dos tokens. Após analisar o arquivo de entrada, é impresso na tela a mensagem léxica correspondente aos tokens reconhecidos ou a mensagem de erro associada caso ocorra um erro. O arquivo de saída contém também todas essas mensagens obtidas na análise.

2. Estrutura de Análise

Foram criadas duas tabelas para facilitar a implementação do autômato finito e do código fonte, uma para o mapeamento de cadeias para tokens que as classificam, e outra que agrupa diferentes classes de cadeias.

a. Tabela de mapeamento cadeia token

As cadeias que podem ser reconhecidas pelo analisador léxico implementado estão mapeadas na tabela 1, em que cada cadeia possui um token (string definido que identifica a classe da cadeia) associado.

Cadeia	Token
program	symb_program
begin	symb_begin
end	symb_end
const	symb_const
var	symb_var
real	type_real
integer	type_integer
procedure	symb_procedure
read	symb_read
write	symb_write
while	symb_while
do	symb_while
for	symb_for
to	symb_to
if	symb_if
then	symb_then
else	symb_else
=	symb_eq
<>	symb_neq
>=	symb_gte
<=	symb_lte
>	symb_gt
<	symb_lt
;	symb_semicol
:	symb_col
,	symb_coma
.	symb_period
:=	symb_assign
+	symb_add
-	symb_diff
*	symb_mult
/	symb_div
a-z A-Z 0-9	ident
0-9	num_int
0-9.0-9	num_real
)	symb_cparentesis

(	symb_oparentesis
{	symb_obrack
}	symb_cbrack

Tabela 1 - Tabela de mapeamento cadeia token.

b. Tabela que agrupa cadeias em conjuntos

A tabela 2 agrupa as classes de cadeias: palavras reservadas, letras, dígitos, caracteres não imprimíveis, assim como possíveis finais para cadeias que geram identificadores, palavras reservadas e números (mais detalhes na seção 3 - Autômato Finito), e os caracteres inválidos da linguagem - todos os outros que não foram mapeados.

Conjunto	Cadeia
palavras reservadas	program, begin, end, const, var, real, integer, procedure, read, write, while, do, for, to, if, then, else
possíveis finais	`= <> >= <= > < ; : , . := + - * / ( ) { }`
caracteres não imprimíveis	espaço (" "), tabulação ("\t"), quebra de linha ("\n")
letra	a-z A-Z
dígito	0-9
caracteres inválidos	outros caracteres que não foram mapeados até agora (como @, #, $, %, &, ç, ...)

Tabela 2 - Tabela que agrupa cadeias em conjuntos.

3. Autômato Finito

O autômato finito implementado para a análise léxica está representado na Figura 1, o qual explicita todos os estados finais possíveis e seus respectivos retornos. Em vários casos, o autômato chega em um estado final quando encontra um caractere "outro" ou "possíveis finais", que representam o começo da próxima cadeia, não fazendo parte da cadeia que está sendo lida. Dessa forma, a cadeia de caracteres até “outro” ou “possíveis finais” é devolvido, e a estratégia de “retroceder” é utilizada para para voltar uma posição de leitura, e o caractere "outro" ou "possíveis finais" lido é o primeiro caractere da nova cadeia a ser lido.

Ademais, no caso de identificadores, palavras reservadas e números, os "possíveis finais" variam de acordo com a porção do código que a cadeia se encontra. Por exemplo, na declaração de um identificador (fat:=1;), o caractere ":" é aceito depois do identificador, mas "*" não. Já quando é realizada uma operação com o identificador fat (fat*aux;), o caractere "*" que era proibido no último caso, pode vir depois de *fat*. Desse modo, nem sempre todos os "possíveis finais" são válidos, de forma que essa checagem extra deve ser realizada no

analisador sintático, por ter que considerar o contexto da cadeia, estando fora do escopo do analisador léxico.

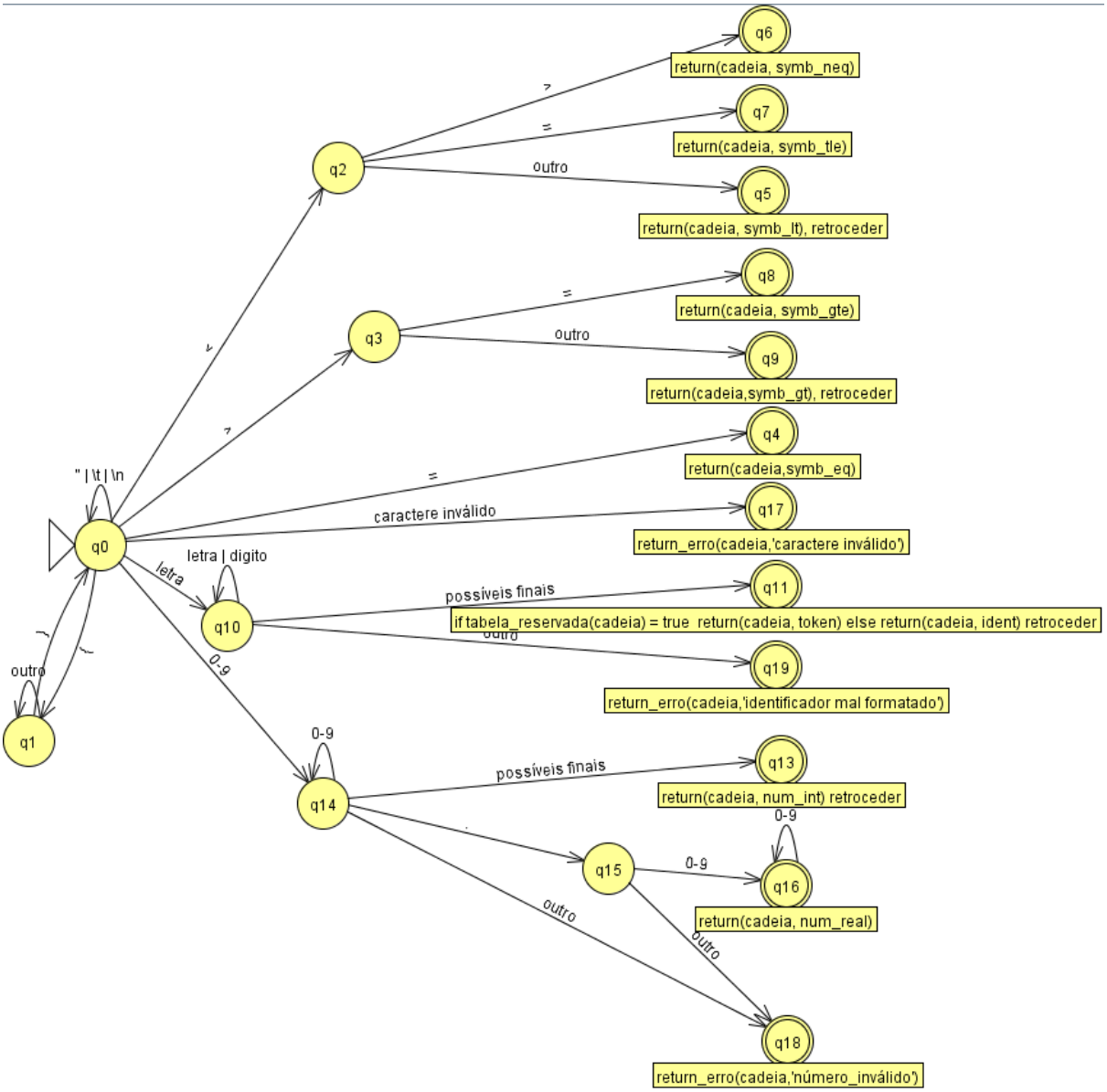

Figura 1 - Autômato finito do analisador léxico.

4. Tratamento de Erros

Quando o analisador léxico reconhece algum erro, ele retorna a cadeia em que o erro foi encontrado. Os seguintes tipos de erros são reconhecidos:

- **Identificador inválido**: ao percorrer cadeias com letras (com ou sem dígitos) que podem representar identificadores ou palavras reservadas, é lido algum caractere que não seja letra, dígito ou possíveis finais (caracteres que podem vir depois de um identificador ou palavra reservada)

- **Número inválido:** ao percorrer cadeias que podem representar números reais ou inteiros, é lido algum caractere que não seja dígito ou, no caso do estado q14, algo diferente de "." ou possíveis finais (caracteres que podem vir depois de um número)
- **Caractere inválido**: quando um caractere inválido é lido no começo de uma cadeia

O erro causado por comentário não fechado, que resulta em fim de arquivo não esperado, é tratado na implementação, pois a cadeia de caracteres terminará e o autômato não está em um estado final.

5. Exemplo de Execução

Essa seção exemplifica a execução do código analisador léxico.

a. Comandos

Em um computador com Python3 instalado, no diretório em que os códigos e o arquivo de entrada estão, o seguinte comando deve executado no terminal (Linux ou Mac):

```
python3 main.py --input input_file_name --output output_file_name
```

b. Arquivo de entrada

Um exemplo de arquivo de entrada utilizado para teste do código.

```
program p;
{Comentário correto.}
var x@: integer;
begin
    x:=1;
    while (x < 3.) do
        x:=x+1;!
end.
{Erro no comentário.
```

c. Arquivo de saída

Arquivo de saída (o mesmo é impresso no terminal), obtido ao executar o código com a entrada definida acima:

```
program,symb_program
p,ident
;,symb_semicol
var,symb_var
```

```
x@,Erro léxico na linha 3: Identificador inválido
:,symb_col
integer,symb_integer
;,symb_semicol
begin,symb_begin
x,ident
:=,symb_assign
1,num_int
;,symb_semicol
while,symb_while
(,symb_oparentesis
x,ident
<,symb_lt
3.,Erro léxico na linha 6: Número inválido
),symb_cparentesis
do,symb_do
x,ident
:=,symb_assign
x,ident
+,symb_add
1,num_int
;,symb_semicol
!,Erro léxico na linha 7: Caracter inválido
end,symb_end
.,symb_dot
,Erro léxico na linha 9: Comentário não finalizado
```

6. Referências

JFLAP Version 7.1 RELEASED July 27, 2018. Página inicial. Disponível em: <http://www.jflap.org/>. Acesso em: 10 de junho de 2021.

Linguagem P--. Pascal Simplificado. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/6212575/mod_resource/content/0/Linguagem%20P--.pdf/>. Acesso em: 10 de junho de 2021.

Repositório de Aulas da disciplina SCC0605. Disponível em: <https://edisciplinas.usp.br/course/view.php?id=86209&section=2#tabs-tree-start>. Acesso em: 10 de junho de 2021.

Repositório do Github com implementação do código. Disponível em: <https://github.com/jcelias98/Lexical_Analysis_P--/tree/master>. Acesso em: 10 de junho de 2021.