UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE CIÊNCIA TECNOLOGIA E ELETRÔNICA

PROF. FAIMISON RODRIGUES PORTO

JOÃO PAULO MATOS MENDES
PÂMELA ALINY CLETO PAVAN

ANALISADORES LÉXICO E SINTÁTICO

SÃO MATEUS – ES
2022

Parte 1: Gerar Máquina de Moore e Analisador Léxico de expressões com parênteses, colchetes e chaves balanceadas, as quatro operações básicas, números e variáveis. Sendo que parênteses, colchetes e chaves são opcionais, mas se eles aparecerem na mesma expressão eles devem obedecer a seguinte ordem: parênteses (mais internos), colchetes e chaves (mais externos).

1.1 Máquina de Moore

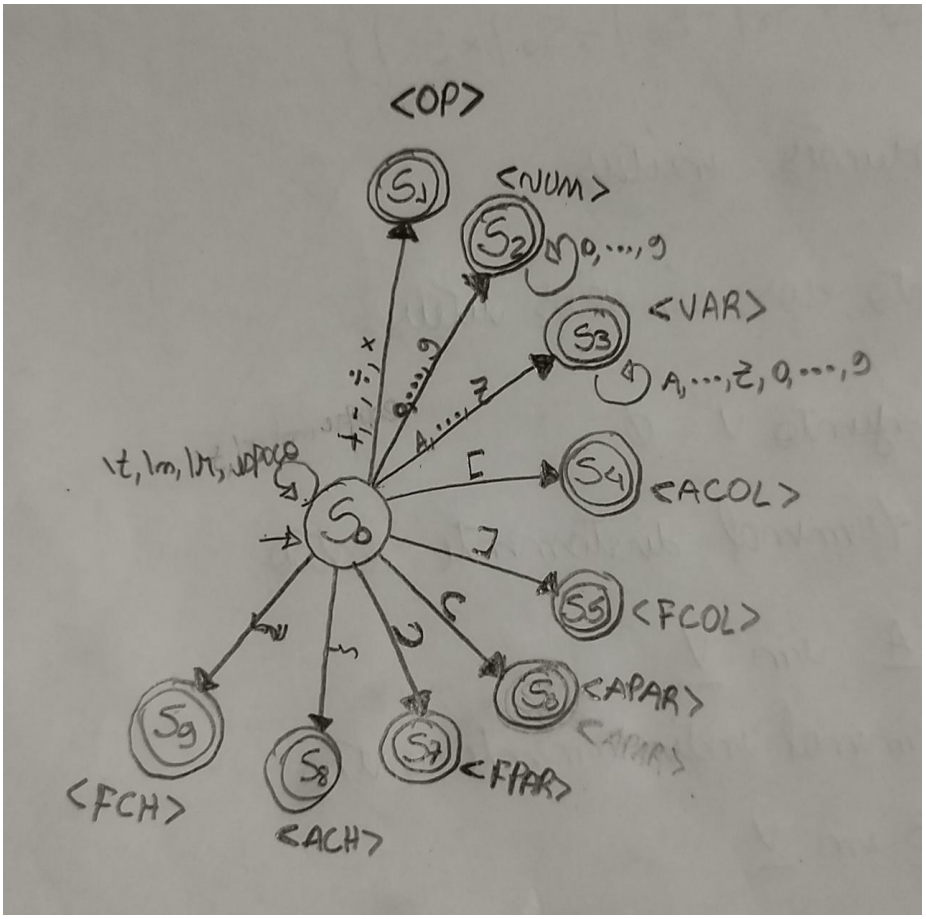

Figura 1 - Máquina de Moore

1.2 Analisador Léxico

Testes do Analisador Léxico - Implementação da Máquina de Moore:
(Execute o arquivo “trabalho_analisadores.py”)

Teste1: (2+2)

Console de Saída:
<ABRE_PARENTESES>
<NUMERO>
<OPERADOR>

<NUMERO>
<FECHA_PARENTESES>
<EOF>

Análisa léxica realizada com sucesso no arquivo 'entrada.txt'

Teste 2: 5*$

Console de Saída:
<NUMERO>
<OPERADOR>
Erro léxico! Caractere encontrado: '$'
Era(m) esperado(s): 'abcdefghijklmnopqrstuvwxyz0123456789+-*/()[]{}'

Teste 3: {[(2+1)*3]}

Console de Saída:
<ABRE_CHAVES>
<ABRE_COLCHETES>
<ABRE_PARENTESES>
<NUMERO>
<OPERADOR>
<NUMERO>
<FECHA_PARENTESES>
<OPERADOR>
<NUMERO>
<FECHA_COLCHETES>
<FECHA_CHAVES>
<EOF>

Análise léxica realizada com sucesso no arquivo 'entrada.txt'

Parte 2: Construir uma gramática livre de contexto que atende a geração da linguagem proposta na Parte 1. Após isso, verificar se a gramática construída atende aos pré-requisitos: fatorada, sem recursividade à esquerda e sem ambiguidade. Em seguida, com a gramática em mãos, atendendo aos pré-requisitos, implementar um analisador sintático.

2.1 Geração Gramática Livre de Contexto: fatorada, sem recursividade à esquerda e sem ambiguidade

Observação: A gramática abaixo já está fatorada, sem recursividade à esquerda e sem ambiguidade.

S -> (S2) A | [S1] A | {S} A | <numero> A | <variavel> A
S1 -> (S2) B | [S1] B | <numero> B | <variavel> B
S2 -> (S2) C | <numero> C | <variavel> C
A -> <op> S A | λ

B -> <op> S1 B | λ
C -> <op> S2 C | λ

2.2 Análise Preditiva

Aplicando a técnica de análise preditiva para geração do algoritmo:

```
função S(){
        se (proxToken==’<variavel>’)
                reconhece (<variavel>); A();

        senão se (proxToken==’<numero>’)
                reconhece (<numero>); A();

        senão se (proxToken==’<abreParenteses>’)
                reconhece(<abreParenteses>);S2(); reconhece(<fechaParenteses>);A();

        senão se (proxToken==’<abreColchetes>’)
                reconhece(<abreColchetes>); S1();reconhece(<fechaColchetes>)A();

        senão se (proxToken==’abreChave’)
                 reconhece(<abreChave>); S();reconhece(<fechaChave>);A();

        senão
                  imprime ("erro"); // era esperado um dos seguintes caracteres: <variavel> + <numero>
                  + <abreParenteses> + <abreColchetes> + <abreChave>
}

função S1(){
        se (proxToken==’<variavel>’)
                    reconhece (<variavel>); B();

        senão se (proxToken==’<numero>’)
                    reconhece (<numero>); B();

        senão se (proxToken=’<abreColchetes>’)
                  reconhece(<abreColchetes>); S1();reconhece(<fechaColchetes>);B();

        senão se (proxToken==’<abreParenteses’)
                  reconhece(<abreParenteses>); S2();reconhece(<fechaParenteses>);B();

        senão
                  imprime ("erro"); // era esperado um dos seguintes caracteres: <variavel> + <numero>
                  + <abreParenteses> + <abreColchetes>

}

função S2(){
```

```
        se (proxToken==’<variavel>’)
                reconhece (<variavel>); C();

        senão se (proxToken==’<numero>’)
                reconhece (<numero>); C();

        senão se (proxToken==’<abreParenteses>’)
                reconhece(<abreParenteses>); S2(); reconhece(<fechaParenteses>);C();

        senão
                imprime ("erro"); // era esperado um dos seguintes caracteres: <variavel> + <numero>
                + <abreParenteses>
}

função A(){
        se (proxToken==’<operador>’)
                reconhece (<operador>); S();A();

        senão
                ;
}

Função B(){
        se (proxToken==’<operador>’)
                reconhece (<operador>); S1();B();

        senão
                ;
}
Função C(){
        se (proxToken==’<operador>’)
                reconhece (<operador>); S2();C();
        senão
                ;
}
```

Obs.: A função "reconhece" tem o objetivo de verificar se o próximo token é de fato o que foi passado como argumento. Se sim então lê o próximo token põe em próxToken, senão dispara uma mensagem de erro.

```
função reconhece (token c) {
          se (próxToken == c)
                  próxToken= lêPróximoTokenDoArquivo();
          senão
                  imprime ("erro"); // era esperado o token representado por c
                  sai_do_programa;
}
```

2.3 Analisadores Léxico e Sintático Integrados

Teste Analisadores Léxico e Sintático Integrados
(Execute o arquivo “AnalisadorLexico_Sintatico”)

Teste1: {w-[3-5]+{4-(x-((y)))-1}-9}-8

Console de Saída:
<ABRE_CHAVES>
<VARIAVEL>
<OPERADOR>
<ABRE_COLCHETES>
<NUMERO>
<OPERADOR>
<NUMERO>
<FECHA_COLCHETES>
<OPERADOR>
<ABRE_CHAVES>
<NUMERO>
<OPERADOR>
<ABRE_PARENTESES>
<VARIAVEL>
<OPERADOR>
<ABRE_PARENTESES>
<ABRE_PARENTESES>
<VARIAVEL>
<FECHA_PARENTESES>
<FECHA_PARENTESES>
<FECHA_PARENTESES>
<OPERADOR>
<NUMERO>
<FECHA_CHAVES>
<OPERADOR>
<NUMERO>
<FECHA_CHAVES>
<OPERADOR>
<NUMERO>
<EOF>

Análisa léxica realizada com sucesso no arquivo 'entrada.txt'
Análisa sintática realizada com sucesso no arquivo 'entrada.txt'

Teste2: ({[(2)]})

Console de Saída:
<ABRE_PARENTESES>
<ABRE_CHAVES>
<ABRE_COLCHETES>
<ABRE_PARENTESES>

<NUMERO>
<FECHA_PARENTESES>
<FECHA_COLCHETES>
<FECHA_CHAVES>
<FECHA_PARENTESES>
<EOF>

Análisa léxica realizada com sucesso no arquivo 'entrada.txt'
Erro sintático! Token encontrado: <ABRE_CHAVES>
Era(m) esperado(s): <VARIAVEL> <NUMERO> <ABRE_PARENTESES>
>>>

Teste 3:[2-{x-5}]

Console de Saída:
<ABRE_COLCHETES>
<NUMERO>
<OPERADOR>
<ABRE_CHAVES>
<VARIAVEL>
<OPERADOR>
<NUMERO>
<FECHA_CHAVES>
<FECHA_COLCHETES>
<EOF>

Análisa léxica realizada com sucesso no arquivo 'entrada.txt'
Erro sintático! Token encontrado: <ABRE_CHAVES>
Era(m) esperado(s): <VARIAVEL> <NUMERO> <ABRE_PARENTESES> <ABRE_COLCHETES>