

**LINGUAGENS FORMAIS:**

Especificação e Implementação de um Analisador Léxico-Sintático

Rafael Silveira, Pablo Margreff e Letícia Sampaio

{rsilveira,pmargreff,lsampaio}@inf.ufpel.edu.br

**1. INTRODUÇÃO**

O Análisador léxico é o programa encarregado do processo de analisar a entrada de caracteres e produzir uma seqüência de símbolos chamado "símbolos léxicos" (tokens), que podem ser manipulados mais facilmente pelo analisador sintático.

Já o Analisador Sintático (*parsing*) é o programa encarregado do processo de analisar uma sequência de simbolos de entrada (obtidos do léxico) para determinar sua estrutura gramatical segundo uma determinada gramática formal. Essa análise faz parte de um compilador, junto com o análisador léxico e semântico.

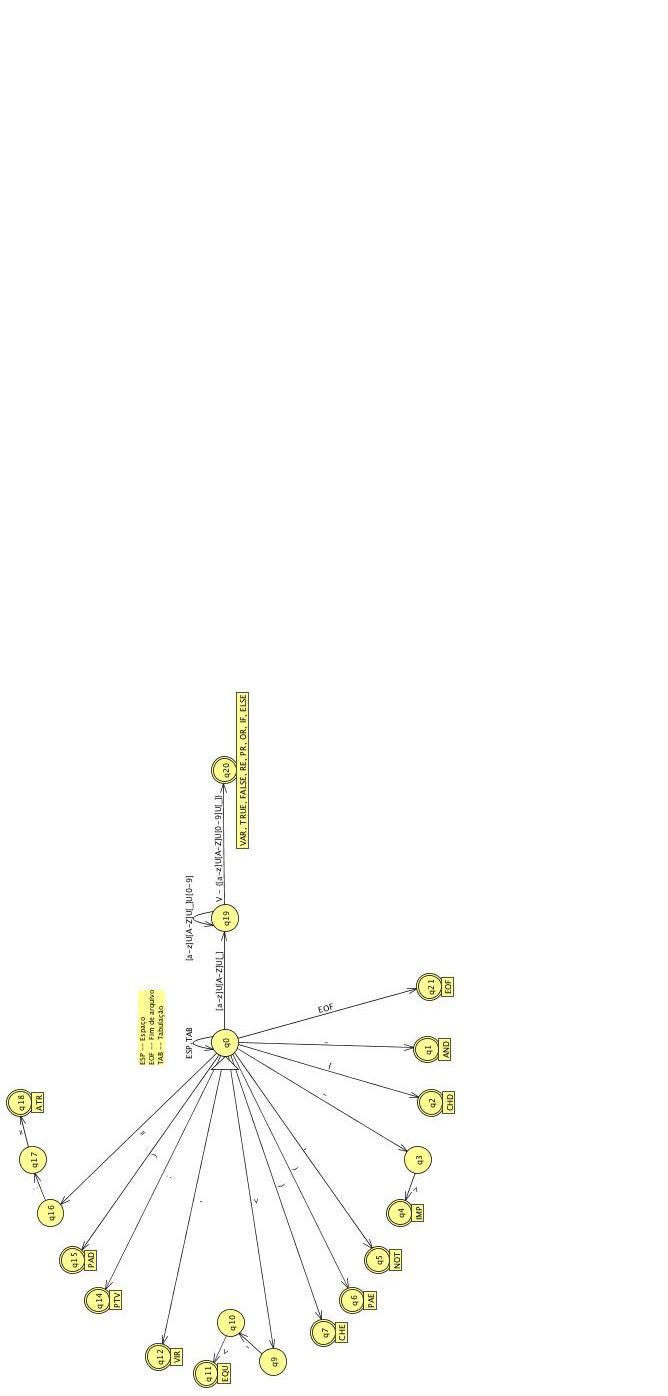
**2. METODOLOGIA**

Primeiramente, foi desenvolvido o autômato finito com uso da ferramenta Jflap, o qual especifica o analisador léxico de acordo com a linguagem solicitada para este trabalho. Logo após, foi escolhida a linguagem de programação Python1, por conta de sua alta abstração e baixa curva de aprendizagem. Após a escolha da linguagem de programação, foi transcrito todo o autômato finito para código, com auxílio da ferramenta de controle de versão Git.

Ao fim do desenvolvimento da primeira parte do trabalho, inciou-se os testes com diversos tipos de entradas de dados. A segunda parte do trabalho, foi desenvolver a gramática livre de contexto, que é a principal parte do analisador sintático. Por fim, foram integrados e testados os códigos das duas partes do trabalho e criada uma interface gráfica, ilustrado na Figura 1.

[[1]](#footnote-0)

**3. AUTÔMATO FINITO**



**3. TABELA DE EQUIVALÊNCIA**

|  |  |  |
| --- | --- | --- |
| **SÍMBOLO** | **SIGNIFICADO** | **TOKEN** |
| ‘ | Negação | NOT |
| ^ | Conjunção | AND |
| v | Disjunção | OR |
| -> | Implicação | IMP |
| <-> | Equivalência | EQU |
| =:= | Atribuição | ATR |
| , | Vírgula | VIR |
| { | Chave Esquerda | CHE |
| } | Chave Direita | CHD |
| Print | Print | PRI |
| Read | Read | REA |
| EOF | Fim de Arquivo | EOF |
| ( | Parênteses Esquerdo | PAE |
| ) | Parênteses Direito | PAD |
| ; | Ponto e Vírgula | PTV |
| [0] U [F] U [False] | Falso | FAL |
| [1] U [T] U [True] | Verdadeiro | TRU |
| [a-z] U [A-Z] U [0-9] | Variável | VAR |

**4. GRAMÁTICA LIVRE DE CONTEXTO**

G = (A,B,P,S).

A = { EXP, V1, V2, SE, SEN, COM }.

B = { VAR, NOT, AND, OR, IMP, EQU, ATR, VIR, CHE, CHD, PRI, REA, EOF, PAE, PAD, PTV, FAL, TRU, IF, ELS }.

P= (

S → VAR ATR EXP PTV (Atribuição)

S → PRI EXP PTV (Print)

S → READ V1 PTV (Read)

S → SE (If)

(Expressão)

EXP → CHD EXP CHD

EXP → EXP EQU EXP

EXP → EXP IMP EXP

EXP → EXP OR EXP

EXP → EXP AND EXP

EXP → V1

EXP → NOT EXP

EXP → TRU

EXP → FAL

(Variável)

V1 → V2

V1 → N NOT V1

V1 → VAR

N → N NOT

N → ⋲

(Variável com virgula)

V2 → V1 VIR V1

(Condicional - SE)

SE → IF PE EXP PD CHE COM CHD SEN

(Condicional - ELSE)

SEN → ELS CHE COM CHD

SEN → ⋲

(Comandos)

COM → EXP PTV COM

COM → SE COM

COM → VAR

COM → VAR ATR EXP COM

COM → REA V1 COM

COM → PRI EXP COM

COM → ⋲

)

1. ***Python*** é uma linguagem de programação de alto nível,, interpretada, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991.Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos Python Software Foundation. [↑](#footnote-ref-0)