|  |  |
| --- | --- |
|  | ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO  Departamento de Engenharia de Computação e Sistemas Digitais |

**Relatório P2**

**Compilador**

**PCS2056 – Linguagens e Compiladores**

|  |  |
| --- | --- |
| Bruno Umeda Grisi | 5438011 |
| Nathalia Sautchuk Patrício | 5432596 |

Índice

[1. Definição da Linguagem 3](#_Toc247476202)

[1.1 Notação BNF 3](#_Toc247476203)

[1.2 Notação de Wirth 4](#_Toc247476204)

[2. Análise Léxica 5](#_Toc247476205)

[3. Análise Sintática 6](#_Toc247476206)

[3.1 Submáquina Program 6](#_Toc247476207)

[3.2 Submáquina Expr 6](#_Toc247476208)

[4. Parte 3 8](#_Toc247476209)

# Definição da Linguagem

A linguagem **LazyComb** é uma combinação de quatro linguagens: Combinator Calculus (CC), Unlambda, Iota e Jot.

## Notação BNF

Abaixo temos a linguagem em notação BNF:

<Program> ::= <CCExpr>

<CCExpr> ::= <CCExpr> <Expr> | epsilon

<Expr> ::= i | <Expr'>

<IotaExpr> ::= i | <Expr'>

<Expr'> ::= I

| K | k

| S | s

| <NonemptyJotExpr

| ` <Expr1> <Expr2>

| \* <IotaExpr1> <IotaExpr2>

| ( <CCExpr> )

<NonemptyJotExpr> ::= <JotExpr> 0

| <JotExpr> 1

<JotExpr> ::= <NonemptyJotExpr> | epsilon

## Notação de Wirth

A partir da notação BNF, criamos a descrição em notação de Wirth abaixo:

Notação de Wirth***:***

Program = CCExpr .

CCExpr = { Expr } .

Expr = "i" | Expr’ .

IotaExpr = "i" | Expr’ .

Expr’ = "I" | "K" | "S" | "k" | "s" | NonemptyJotExpr | "‘" Expr Expr

| "\*" IotaExpr IotaExpr | "(" CCExpr ")" .

NonemptyJotExpr = JotExpr ( "0" | "1" ) .

JotExpr = [ NonemptyJotExpr ] .

Notação de Wirth simplificada:

Program = { Expr }.

Expr = "i" | "I" | "K" | "k" | "S" | "s" | ( "0" | "1") { "0" | "1" } | "`" Expr Expr | "\*" Expr Expr | "(" { Expr } ")" .

Notação de Wirth com atenção aos tipos de linguagem que compõe a ***Lazy-Comb:***

Program = { Expr } . #Lazy-Comb

Expr = "i" #CC | Iota

| "I" #CC

| "K" | "k" #CC

| "S" | "s". #CC

| ( "0" | "1") { "0" | "1" } #Jot

| "`" Expr Expr #CC

| "\*" Expr Expr #Iota

| "(" { Expr } ")" . #Lazy-Comb

# Análise Léxica

O analisador léxico da linguagem **LazyComb** é simples. Todos os tokens pertencentes a seu Léxico são palavras reservadas e está representado pela Figura 1.



**Figura 1 – Analisador Léxico**

Sua simplicidade se dá porque todos os tokens são constituídos de apenas um caracter. Além disso, todos eles são previamente definidos.

# Análise Sintática

Nesta seção, foram projetados os Autômatos Estruturados de Pilha com a finalidade de reconhecer a linguagem **LazyComb** .

## Submáquina Program

|  |  |
| --- | --- |
| **Program** = | ⇟ {⇟ **Expr** ⇟ } ⇟ .  0 1 2 1 |

## Submáquina Expr

|  |  |
| --- | --- |
| **Expr** = | ⇟ **"i"** ⇟ | ⇟ **"I"** ⇟ | ⇟ **"K"** ⇟ | ⇟ **"k"** ⇟ | ⇟ **"S"** ⇟ | ⇟ **"s"** ⇟ | ⇟ ( ⇟ **"0"** ⇟ | ⇟ **"1"** ⇟ ) ⇟  0 1 0 2 0 3 0 4 0 5 0 6 0 0 8 0 9 7  { ⇟ **"0"** ⇟ | ⇟ **"1"** ⇟ } ⇟ | ⇟ **"`"** ⇟ **Expr** ⇟ **Expr** ⇟ | ⇟ **"\*"** ⇟ **Expr** ⇟ **Expr**  ⇟ | ⇟  10 11 10 12 10 0 13 14 15 0 16 17 18 0  **"("** ⇟ { ⇟ **Expr**  ⇟ } ⇟ **")"**  ⇟ .  19 20 21 20 22 |

Reservando os direitos autorais ao metacompilador desenvolvido pela **Compilex®,** foram gerados os Autômatos Estruturados de Pilha para cada uma das submáquinas anteriores e estão representados pelas Figuras 2 e 3.



**Figura 2 – Autômato Estruturado de Pilha da Submáquina *Program***



**Figura 3 – Autômato Finito Determinístico da Submáquina *Expr***

# Parte 3