



ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO
PAULO

Linguagens e Compiladores

Relatório P2

**Henrique Sussumu Matsui Kano
Mi Che Li Lee**

1 de dezembro de 2015

Sumário

Sumário	i
1 Construção do reconhecedor	1
1.1 Notação BNF	1
1.2 Notação Wirth	2
1.3 Análise Léxica	2
1.4 Análise Sintática	3

Capítulo 1

Construção do reconhecedor

Nessa etapa foi feita a construção do reconhecedor para a linguagem LazyComb baseado no autômato de pilha estruturado.

1.1 Notação BNF

Linguagem em notação BNF:

```
<Program> ::= <CCEpr>

<CCEpr> ::= <CCEpr> <Expr> | epsilon

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

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

<Expr'> ::= I
          | K | k
          | S | s
          | <NonemptyJotExpr>
          | ' <Expr1> <Expr2>
          | * <IotaExpr1> <IotaExpr2>
          | ( <CCEpr> )

<NonemptyJotExpr> ::= <JotExpr> 0
                   | <JotExpr> 1

<JotExpr> ::= <NonemptyJotExpr> | epsilon
```

1.2 Notação Wirth

A partir da notação em BNF, foi obtida a notação em Wirth:

```
Program      = CCEXpr .

CCEXpr       = { Expr }.

Expr         = "i" | Expr '

IotaExpr     = "i" | Expr '

Expr '       ::= "I"
                | "K" | "k"
                | "S" | "s"
                | NonemptyJotExpr
                | "'" Expr Expr
                | "*" IotaExpr IotaExpr
                | "(" CCEXpr ")"

NonemptyJotExpr ::= JotExpr "0"
                  | JotExpr "1"

JotExpr = NonemptyJotExpr | epsilon.
```

1.3 Análise Léxica

Como todos os terminais da linguagem são compostos apenas por um caracter, a tarefa do analisador léxico se torna simples, basta verificar se a entrada está contida no conjunto de caracteres válidos.

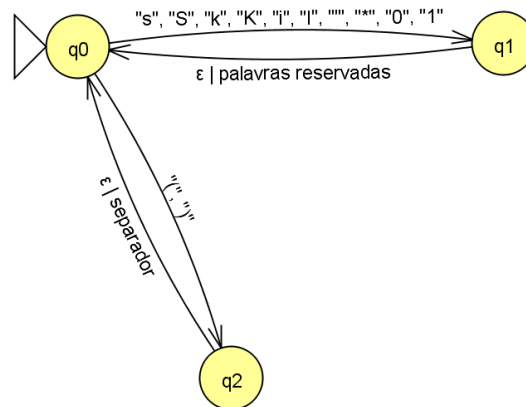


Figura 1.1: Transdutor - Analisador léxico

1.4 Análise Sintática

A partir da notação em Wirth, foi feita uma simplificação:

```
Program = { Expr }.
```

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

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

Para cada não terminal foi gerada uma máquina sintática, representada pelos seu estado inicial, estados finais e transições entre estados. Cada transição pode se manifestar na forma de chamada ou retorno para uma submáquina, ou execução de uma ação semântica, tudo isso usando autômato de pilha estruturado (empilhando ou desempilhando máquinas).

Utilizando o site <http://mc-barau.herokuapp.com/> e o programa JFLAP, a descrição da linguagem em notação de Wirth resultou nas seguintes máquinas:

PROGRAM

```
Program = 0 { 1 Expr 2 } 1 .
```

```
Program:
Minimized DFA:
initial: 0
final: 1
(0, Expr) -> 1
(1, Expr) -> 1
```

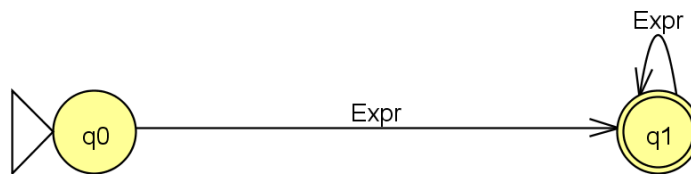


Figura 1.2: Program

EXPR

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

```
Sub-maquina Expr:
Minimized DFA:
initial: 0
final: 1, 2
(0, "i") -> 1
(0, "I") -> 1
(0, "K") -> 1
(0, "k") -> 1
(0, "S") -> 1
(0, "s") -> 1
(0, "0") -> 2
(0, "1") -> 2
```



```
Expr 12 | 0 "*" 13 IotaExpr 14 IotaExpr 15 | 0 "(" 16 { 17  
Expr 18 } 17 ")" 19 .
```

Sub-maquina IotaExpr:

Minimized DFA:

initial: 0

final: 1, 2

(0, "i") -> 1

(0, "I") -> 1

(0, "K") -> 1

(0, "k") -> 1

(0, "S") -> 1

(0, "s") -> 1

(0, "0") -> 2

(0, "1") -> 2

(0, "'") -> 3

(0, "*") -> 4

(0, "(") -> 5

(2, "0") -> 2

(2, "1") -> 2

(3, Expr) -> 7

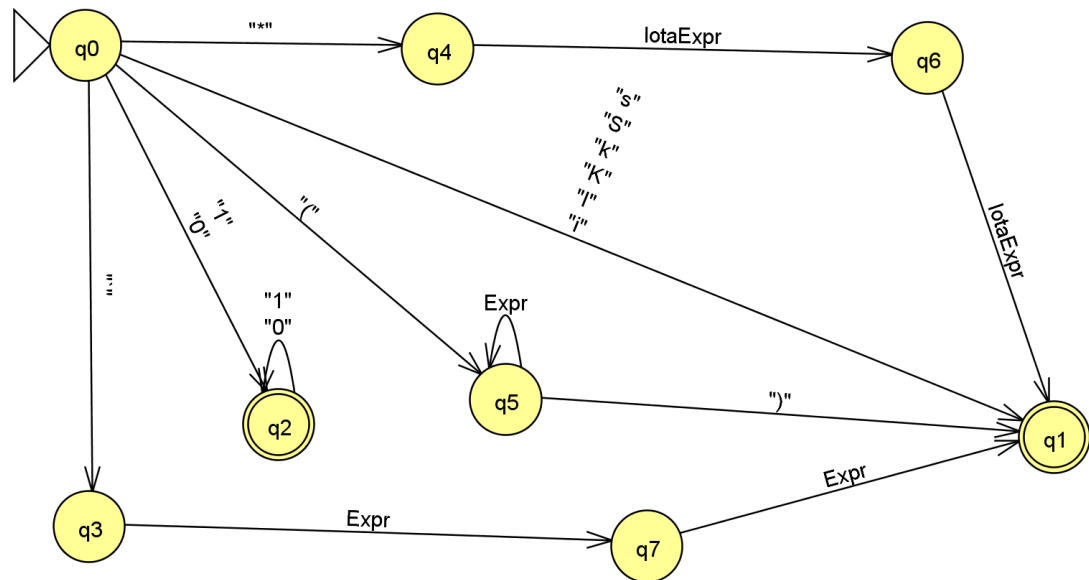
(4, IotaExpr) -> 6

(5, Expr) -> 5

(5, ")") -> 1

(6, IotaExpr) -> 1

(7, Expr) -> 1

Figura 1.4: `lotaExpr`

Referências Bibliográficas

- [1] R. Ricardo S. Jaime A. Reginaldo, B. Anarosa. Introdução máquina de von neumann.
- [2] João José Neto. *Introdução à Compilação*. Escola Politécnica da USP, 1 edition, 1986.