**Teoria da Computação**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Digitado por:** | **Professor:** | **Versão** | **Ano** |
| Ricardo Kim | Pier Richetti | 1.0 | 2017 |

**ÍNDICE**

[1. Apresentação e Ementa 3](#_Toc499129315)

[2. Introdução 4](#_Toc499129316)

[3. Autômato Finito 5](#_Toc499129317)

[3.1. Representação: 5](#_Toc499129318)

[3.2. Exemplo: 5](#_Toc499129319)

[4. Automato Finito Deterministico (AFD) 7](#_Toc499129320)

[4.1. Definição Formal 7](#_Toc499129321)

[4.2. Exercícios 7](#_Toc499129322)

[4.3. Soluções 8](#_Toc499129323)

[4.4. Automomato em C 8](#_Toc499129324)

[4.4.1. Função 8](#_Toc499129325)

[4.4.2. Goto 9](#_Toc499129326)

[4.4.3. Laboratório 10](#_Toc499129327)

[5. Automatos Finitos Não Deterministicos (AFN) 14](#_Toc499129328)

[5.1. Definição formal 14](#_Toc499129329)

[5.2. Exemplo 14](#_Toc499129330)

[5.3. Exercício 14](#_Toc499129331)

[5.3.1. Exemplo AFD 15](#_Toc499129332)

[5.3.2. Exemplo AFN 15](#_Toc499129333)

[6. Conversão AFN para AFD 16](#_Toc499129334)

[6.1. Exemplo 16](#_Toc499129335)

[6.1.1. Passo 1 16](#_Toc499129336)

[6.1.2. Passo 2 16](#_Toc499129337)

[6.1.3. Passo 3 16](#_Toc499129338)

[6.2. Exercício 17](#_Toc499129339)

[7. Expressões regulares 19](#_Toc499129340)

[7.1. Conversão de C.R em A.F 20](#_Toc499129341)

[8. Transdutor 22](#_Toc499129342)

[9. Análise Léxica 24](#_Toc499129343)

[9.1. Transdutor 24](#_Toc499129344)

[10. Gramáticas 27](#_Toc499129345)

[10.1. Exercício 2 28](#_Toc499129346)

[11. Compilador 29](#_Toc499129347)

[11.1. Fases 29](#_Toc499129348)

[11.2. Análise léxica 29](#_Toc499129349)

[11.3. Análise Sintática 29](#_Toc499129350)

[11.4. Análise Semântica 29](#_Toc499129351)

[11.5. Tipos de tradutor 30](#_Toc499129352)

[11.6. Montadores 30](#_Toc499129353)

[11.6.1. Macro montador 30](#_Toc499129354)

[11.7. Analises 30](#_Toc499129355)

[11.7.1. Sintática 30](#_Toc499129356)

[11.7.2. Semântica 30](#_Toc499129357)

# Apresentação e Ementa

* Assuntos
  + Conceitos de linguagens formais.
  + Alfabeto, palavra linguagem.
  + Reconhecedores
  + Expressões Regulares
  + Gramáticas
  + Maquina de Turing
  + Compiladores
  + Analizadores:
    - Léxicos
    - Sintáticos
    - Semânticos
  + Teoria da Computação

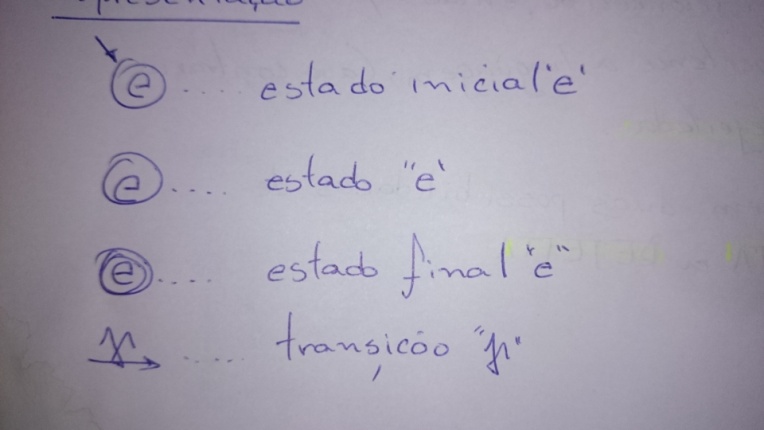
# Introdução

* Alfabeto: Conjunto de Símbolos ex:
  + Σ = {0,1} (Alfabeto binário)
* Palavra: é uma sequência de símbolos formados de um alfabeto, colocado lado a lado e sem vírgulas ex:
  + palavra: 0110 é palavra sobre Σ = {0,1}
* Linguagem: é um conjunto de palavras ex:
  + L = {00, 01, 10, 11 }
  + L = {ω| ω é formado exclusivamente por 2 símbolos } Σ = {0,1}

# Autômato Finito

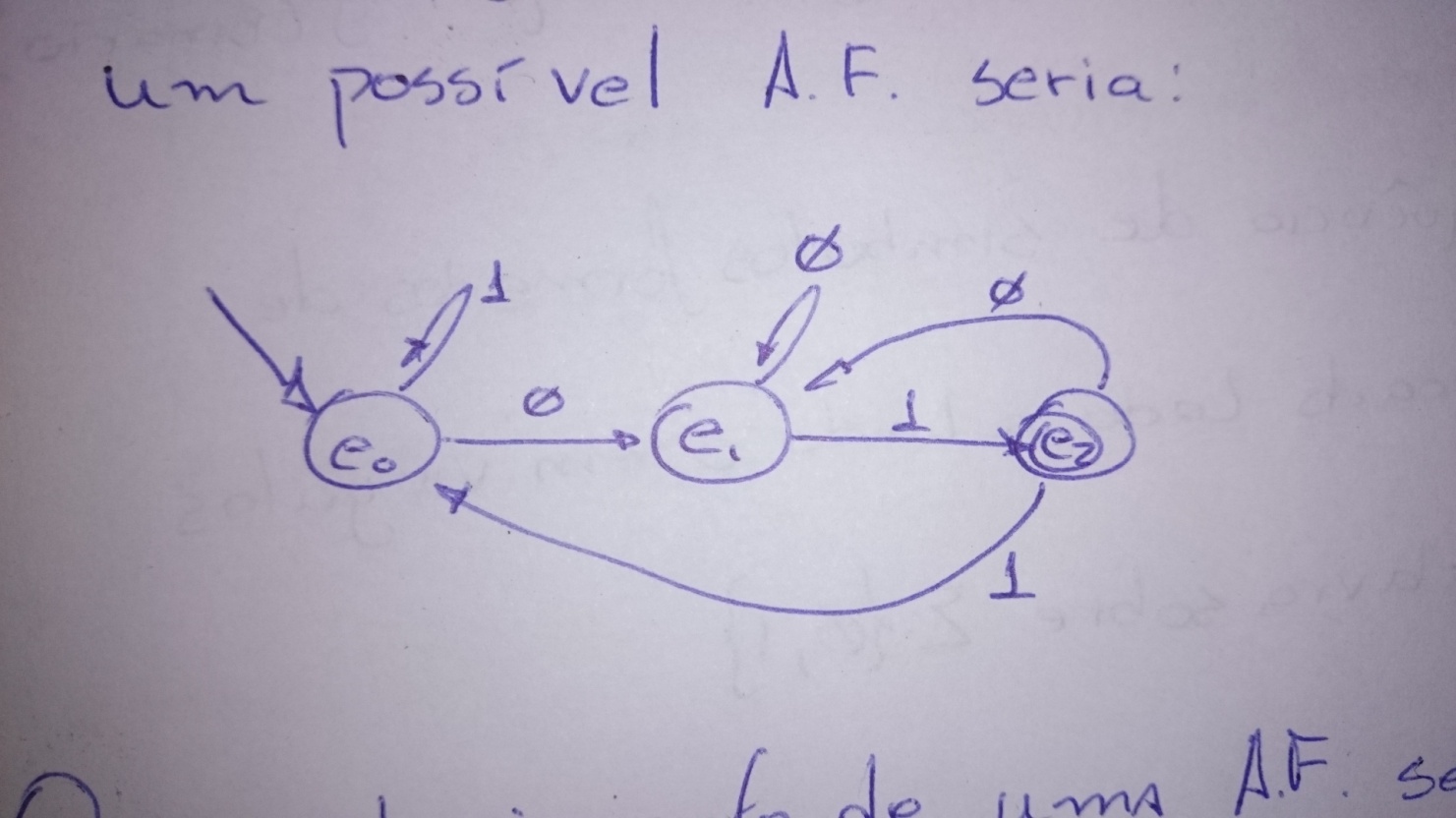
Um AF é uma máquina de estados que pode atuar como **reconhecedor**

## Representação:



## Exemplo:

Para L = {ω| ω finaliza em 01} e sendo Σ = {0,1} um possível AF seria:

[2]

O Reconhecimento de uma AF se dá da seguinte forma:

* A partir do estado inicial e do primeiro símbolo uma **função de transição** é consultada, um novo estado é alcançado e o processo prossegue.
* Se ao final do reconhecimento, o estado for **estado final** e toda a sentença (ou palavra) for reconhecida, então a palavra é dita **Aceita** como pertencente a linguagem. Caso contraria a palavra é **Rejeitada**.
* Um AF só tem duas possibilidades de resposta: **Aceita** ou **Rejeita**.

No exemplo para ω = 0101, teremos:

|  |  |
| --- | --- |
| 0 | e0 🡪 e1 |
| 1 | e1 🡪 e2 |
| 0 | e2 🡪 e1 |
| 1 | e1 🡪 e2 |

Estado é final? 🡪 Sim

A palavra acabou: 🡪 Sim

**Aceita**

No exemplo para ω = 00, teremos:

|  |  |
| --- | --- |
| 0 | e0 🡪 e1 |
| 0 | e1 🡪 e1 |

Estado é final? 🡪 Não

A palavra acabou: 🡪 Sim

**Rejeita**

No exemplo para ω = 012, teremos:

|  |  |
| --- | --- |
| 0 | e0 🡪 e1 |
| 1 | e1 🡪 e2 |
| 2 | e2 🡪 ? |

Estado é final? 🡪 Sim

A palavra acabou: 🡪 Não

**Rejeita**

No exemplo para ω = 02, teremos:

|  |  |
| --- | --- |
| 0 | e0 🡪 e1 |
| 2 | e1 🡪 ? |

Estado é final? 🡪 Não

A palavra acabou: 🡪 Não

**Rejeita**

# Automato Finito Deterministico (AFD)

## Definição Formal

M = (Σ, Q, δ, q0, F)

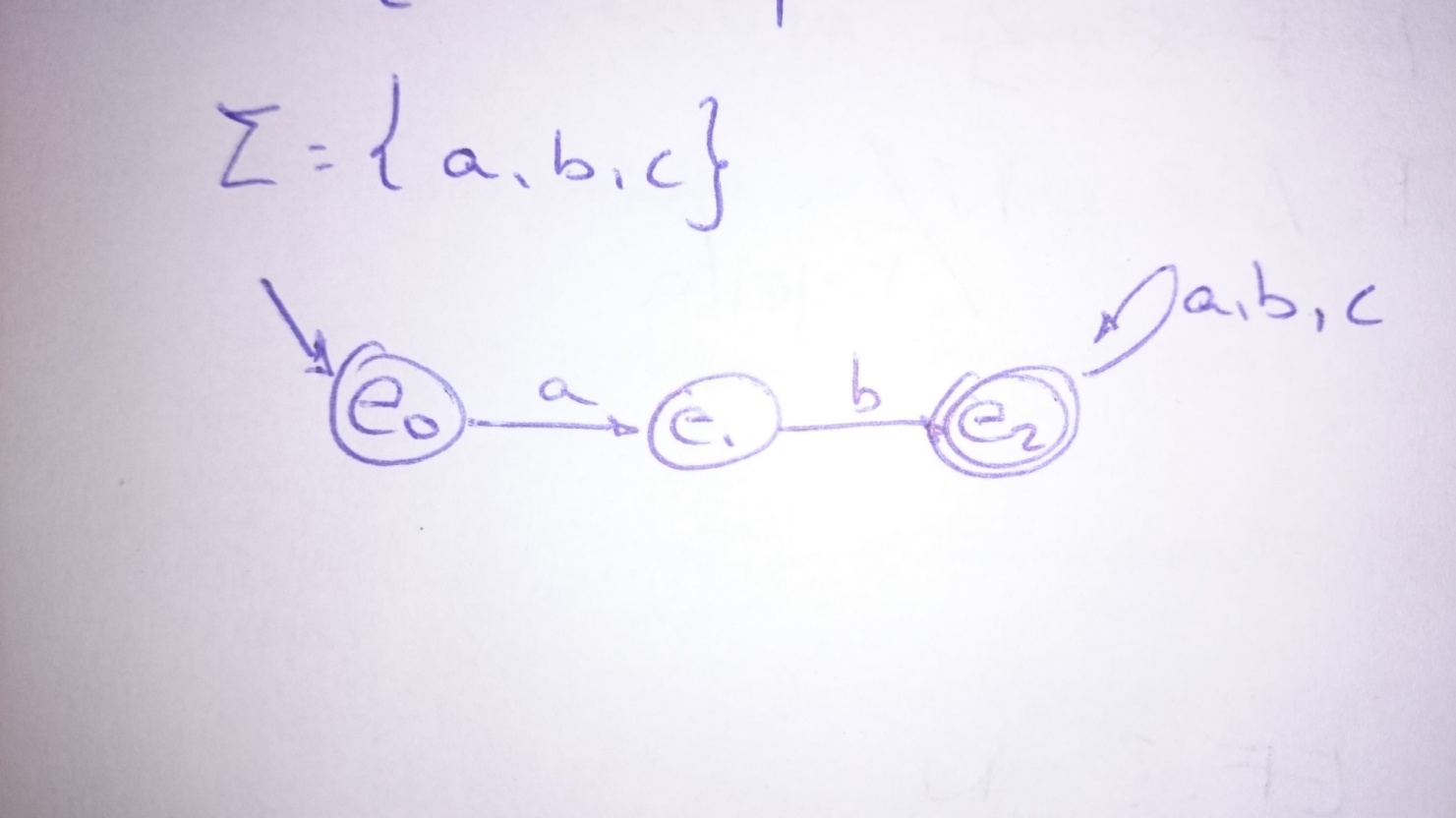
Um AF Determinístico é definido pela quíntupla acima, onde:

|  |  |
| --- | --- |
| Σ | Alfabeto (conjunto) |
| Q | Conjunto de estados |
| δ | Função de Transição |
| q0 | Estado Inicial |
| F | Conjunto de estados finais |

Ex:

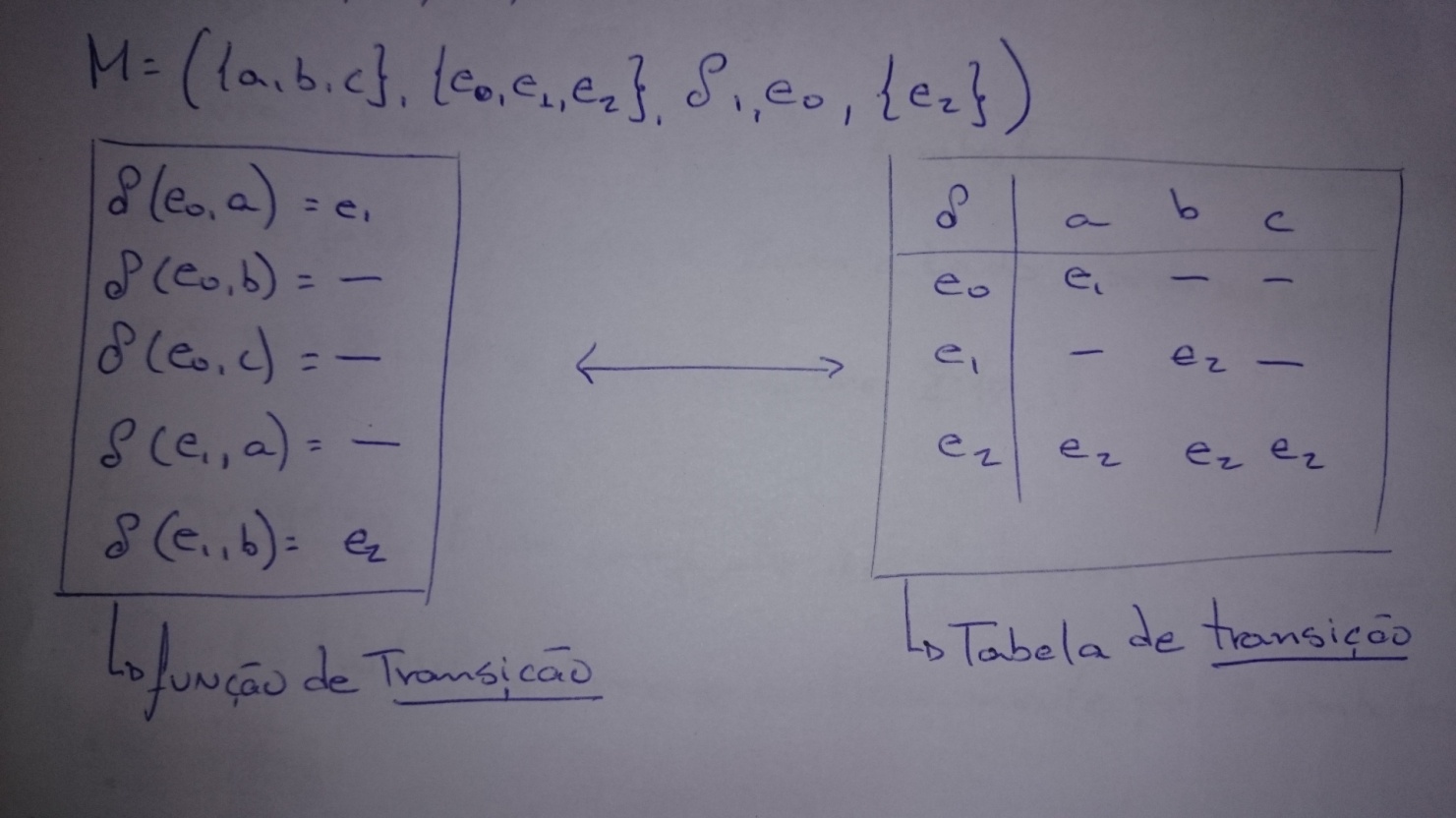
L = { ω | ω possui ab como prefixo}

Σ = {a,b,c}



M = (Σ, Q, δ, q0, F)

M = ({a, b, c} , {e0, e1, e2} , δ1 , e0 , {e2})

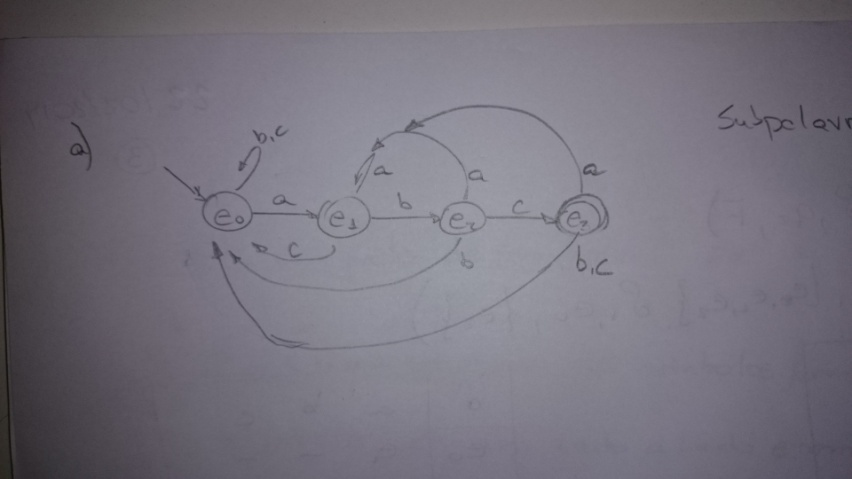


## Exercícios

Sabendo-se que Σ = {a,b,c}, construir reconhecedores para:

1. L = {ω| ω finaliza em abc}
2. L = {ω| ω possui abc como subpalavra}
3. L = {ω| ω possui pelo menos dois símbolos “a”}
4. L = {ω| ω possui exatamente 2 simbolos “a”}
5. L = {ω| ω possui exatamente a sequencia aa}
6. L = {ω| ω possui um número par de a}
7. L = {ω| ω possui um número ímpar de b}
8. L = {ω| ω possui ab ou ca como sufixo}
9. L = {ω| ω possui ab ou ca como subpalavra}
10. L = {ω| ω possui um número par de a e npumero ímpar de b}

## Soluções

1. 

## Automomato em C

### Função

**#include**<stdio.h>

**#include**<stdlib.h>

**#include**<conio.h>

**void** **e0**();

**void** **e1**();

**void** **e2**();

**void** **aceita**();

**void** **rejeita**();

**int** p=0;

**char** f[100];

**int** **main**(**int** argc, **char** \*\*argv) {

p=0;

**puts**("palavra?");

get(f);

e0();

**return** 0;

}

**void** **e0**(){

**if**(f[p] =='a'){

p++;

e1();

}**else**

rejeita();

}

**void** **e1**(){

**if**(f[p]=='b'){

p++;

e0();

}**else**

**if**(f[p]==0)

aceita();

**else**

rejeita();

}

**void** **e2**(){

**if**(f[p]=='a'){

p++;

e0();

}**else**

**if**(f[p]==0)

aceita();

**else**

rejeita();

}

**void** **aceita**(){

**puts**("Aceita");

**getch**();

**exit**(0);

}

**void** **rejeita**(){

**puts**("Rejeita");

**getch**();

**exit**(0);

}

### Goto

**#include**<stdio.h>

**#include**<stdlib.h>

**#include**<conio.h>

**int** **main**(){

**int** p;

**char** f[100];

p=0;

**puts**("palavra?");

**gets**(f);

**goto** e0;

e0:

**if**(f[p]=='a'){

p++;

**goto** e1;

} **else**

**goto** rejeita;

e1:

**if**(f[p]=='b'){

p++;

**goto** e0;

}**else**

**if**(f[p]=='c'){

p++;

**goto** e2;

}**else**

**goto** rejeita;

e2:

**if**(f[p]=='a'){

p++;

**goto** e0;

}**else**

**if**(f[p]==0)

**goto** aceita;

**else**

**goto** rejeita;

aceita:

**puts**("Aceita");

**getch**();

**return**(0);

rejeita:

**puts**("Rejeita");

**getch**();

**return**(0);

**return** 0;

}

### Laboratório

#### Função

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<locale.h>

void e0();

void e1();

void e2();

void aceita();

void rejeita();

void check();

char f[100];

int p;

int main(){

setlocale(LC\_ALL,"portuguese");

p=-1;

puts("Qual é palavra?");

gets(f);

e0();

return 0;

}

void e0(){

p++;

check();

if(f[p]=='a')

e1();

else

rejeita();

}

void e1(){

p++;

check();

if(f[p]=='b')

e0();

else if( f[p]== 'c')

e2();

else if (f[p] == 'a')

e1();

else

rejeita();

}

void e2(){

p++;

check();

if(f[p] == 'a')

e1();

else if(f[p] == '\0')

aceita();

else

rejeita();

}

void aceita(){

puts("Aceita");

getch();

exit (0);

}

void rejeita(){

puts("Rejeita");

getch();

exit (0);

}

void check(){

printf("f[%d] = %c\n",p+1,f[p] );

}

#### Goto

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<locale.h>

char f[100];

int p = -1;

void check(){

printf("f[%d] = %c\n",p+1,f[p] );

}

int main(){

setlocale(LC\_ALL,"portuguese");

puts("Qual é palavra?");

gets(f);

goto e0;

e0:

p++;

check();

if(f[p]=='a')

goto e1;

else

goto rejeita;

e1:

p++;

check();

if(f[p]=='b')

goto e0;

else if( f[p]== 'c')

goto e2;

else if (f[p] == 'a')

goto e1;

else

goto rejeita;

e2:

p++;

check();

if(f[p] == 'a')

goto e1;

else if(f[p] == '\0')

goto aceita;

else

goto rejeita;

aceita:

puts("Aceita");

getch();

return 0;

rejeita:

puts("Rejeita");

getch();

return 0;

return 0;

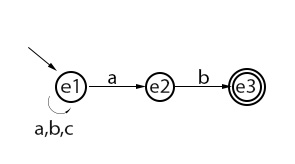
}

# Automatos Finitos Não Deterministicos (AFN)

## Definição formal

Um AFN, a partir de um estado e um símbolo, poderá haver mais de um próximo estado.

## Exemplo



|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Obs: A palavra é dita aceita quando, em pelo menos um dos caminhos houver sucesso.

Reconhecimento:

## Exercício

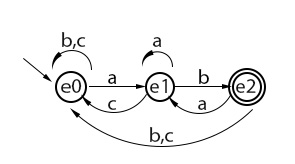
Dado que: construir um AFN para:

Reconhecimento formal de uma palavra por um AFD ou AFN:

Função de transição entendida.

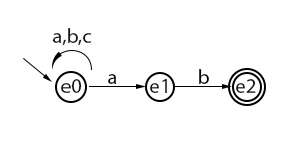
Definimos que:

### Exemplo AFD



Reconhecimento de

### Exemplo AFN

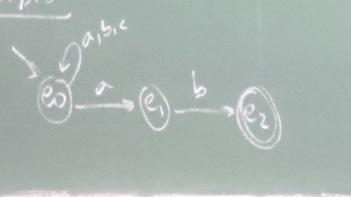


Reconhecimento de

# Conversão AFN para AFD

Para converter um AFN, aplicamos os seguintes passos:

## Exemplo



|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Passo 1

Construir uma tabela especial “”, em que a partir do estado inicial são verificados todos os estados alcançáveis entre “<” e “>” sem vírgulas, cada “tag” deve ser desenvolvida.

No caso do exemplo.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Passo 2

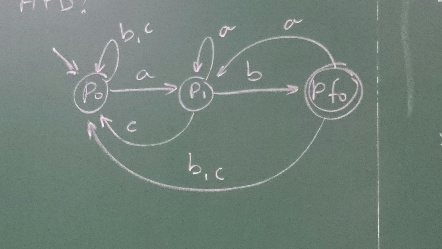
Nomear cada “tag” por cada “tag” que possuir algum estado final irá gerar um estado final

Ou seja

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Passo 3

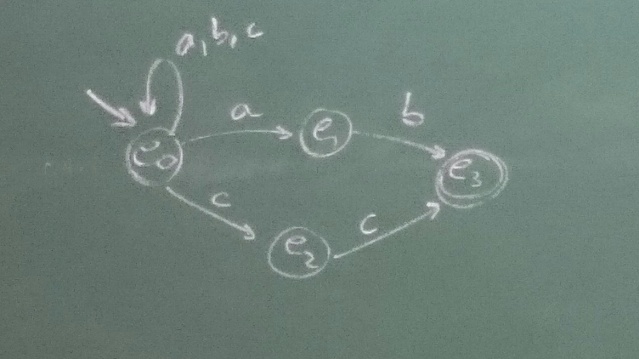
Definir e construir o AFD!



|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Exercício

**Converter em AFD**



**Tabela do AFN**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

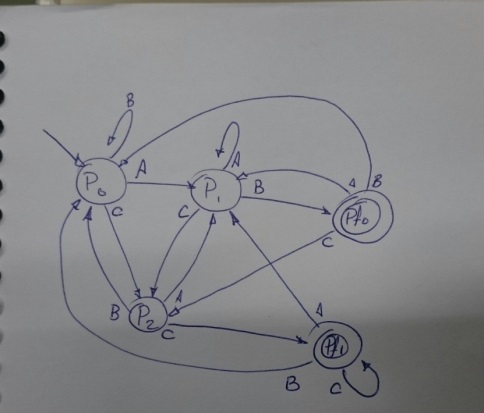
**Algoritmo de Conversão**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Substituição das tag**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**AFD convertido**



|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Expressões regulares

Uma E.R pode ser representada pelos exemplos abaixo:

símbolo

0 ou mais símbolo

uma ocorrência de ou uma ocorrência de

a sequencia

qualquer palavra sobre

obs:

palavras que não contêm dois a consecutivas, sendo

Uma EiR denota e lementos de uma linguagem.

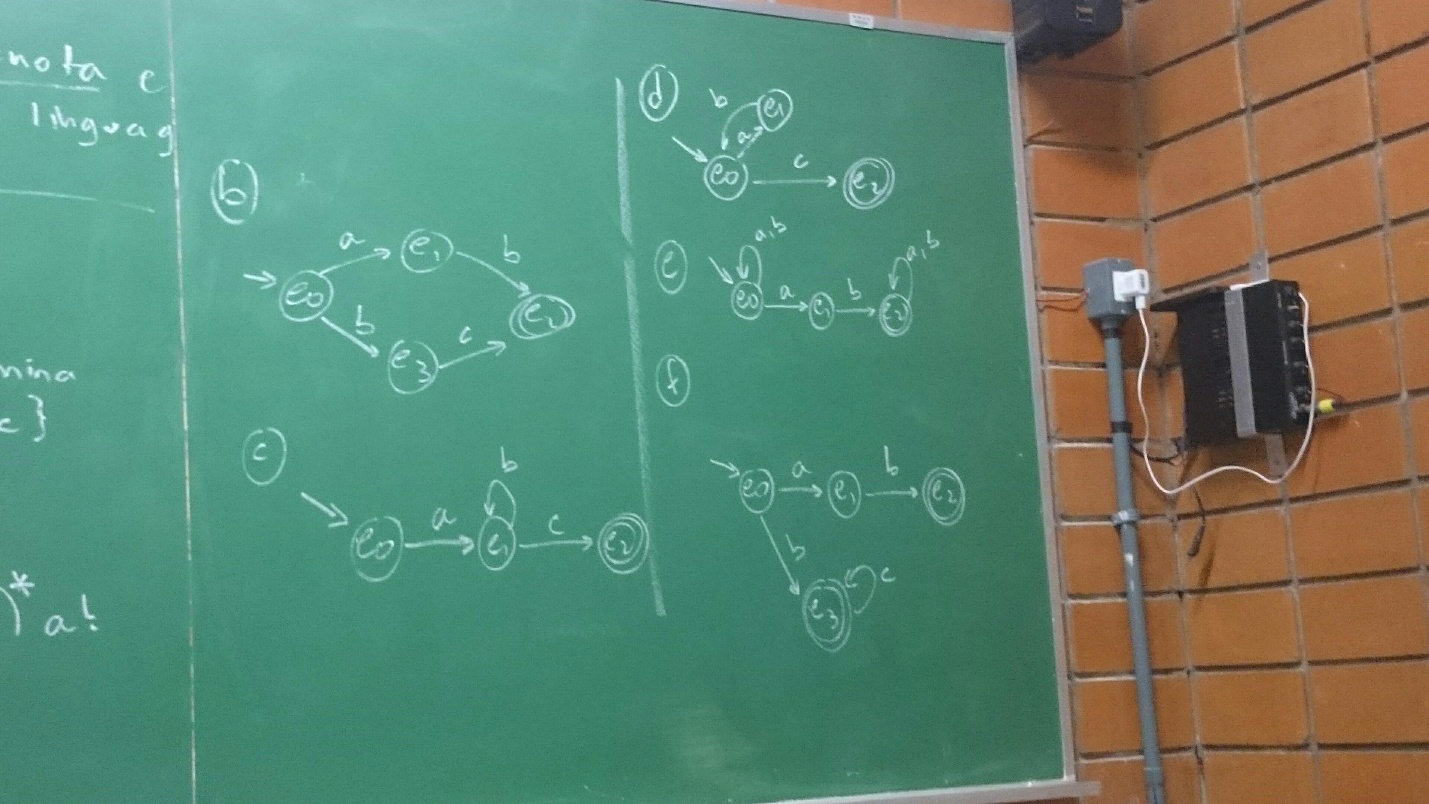
Exemplo:

Exercícios

Dado que:

Construir reconhecedores para as linguagens de notadas por:

Resolução:

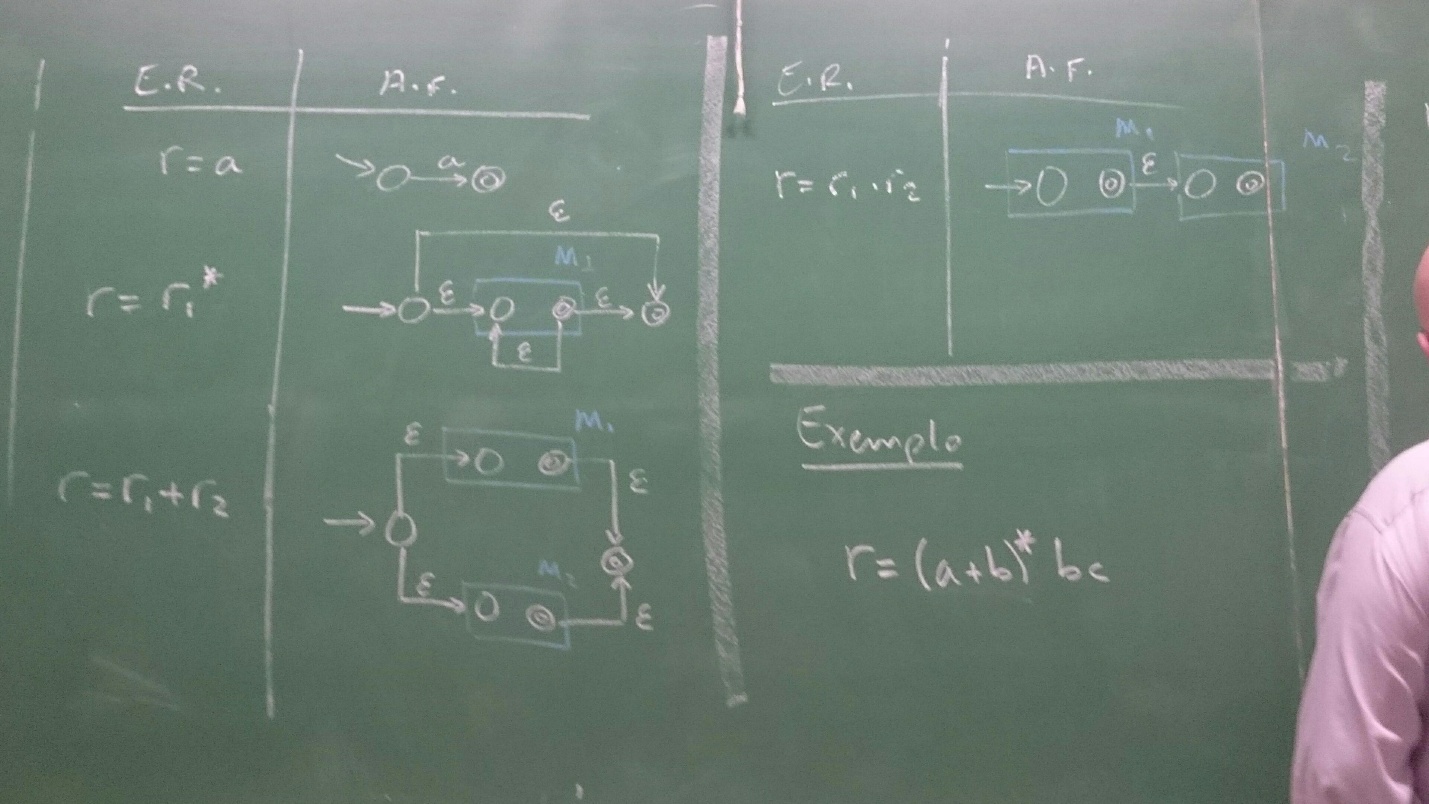


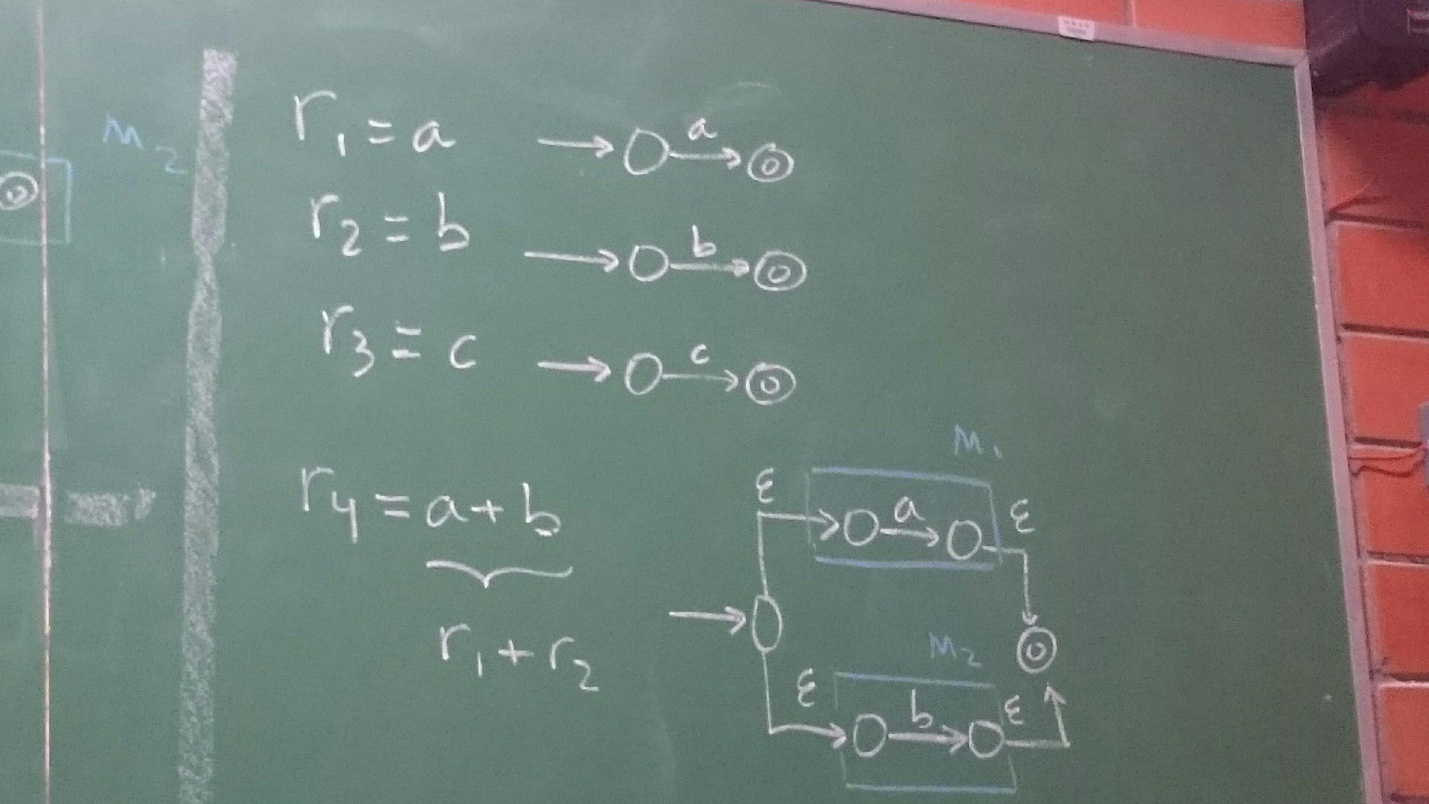
## Conversão de C.R em A.F

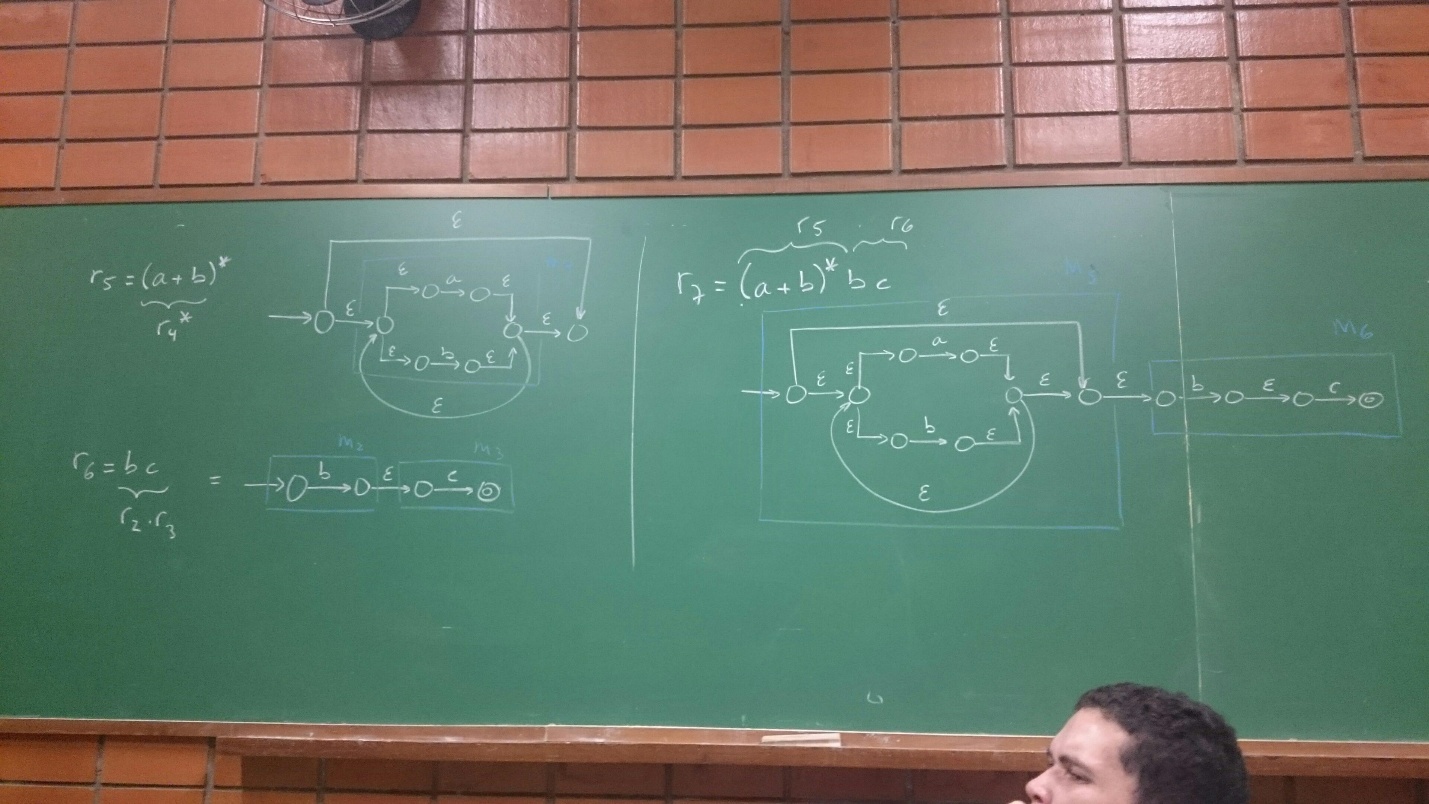
Para converter uma em , usamos a tabela abaixo, considerando-se que:

é um símbolo qualquer São

são máquina que reconhecem respectivamente





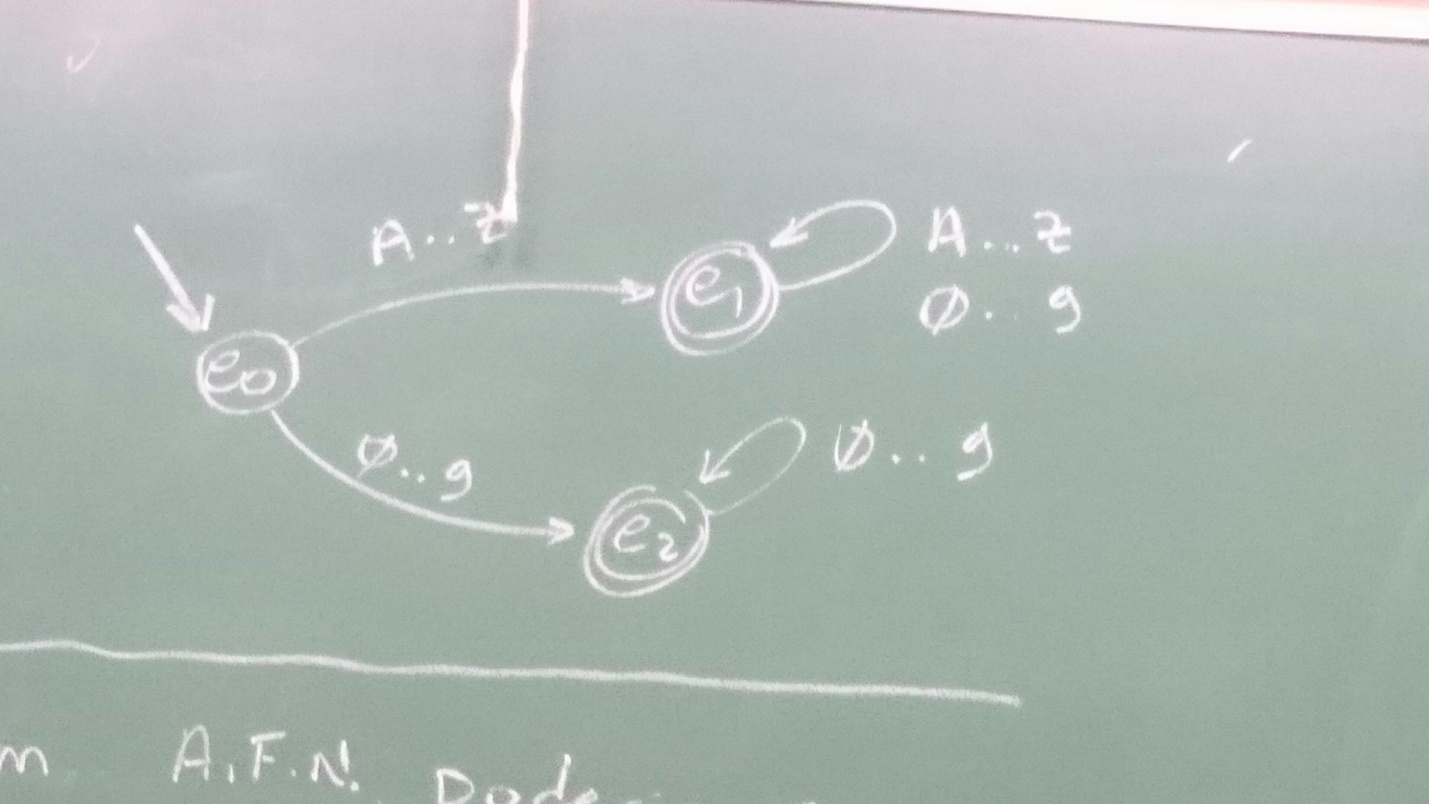


# Transdutor

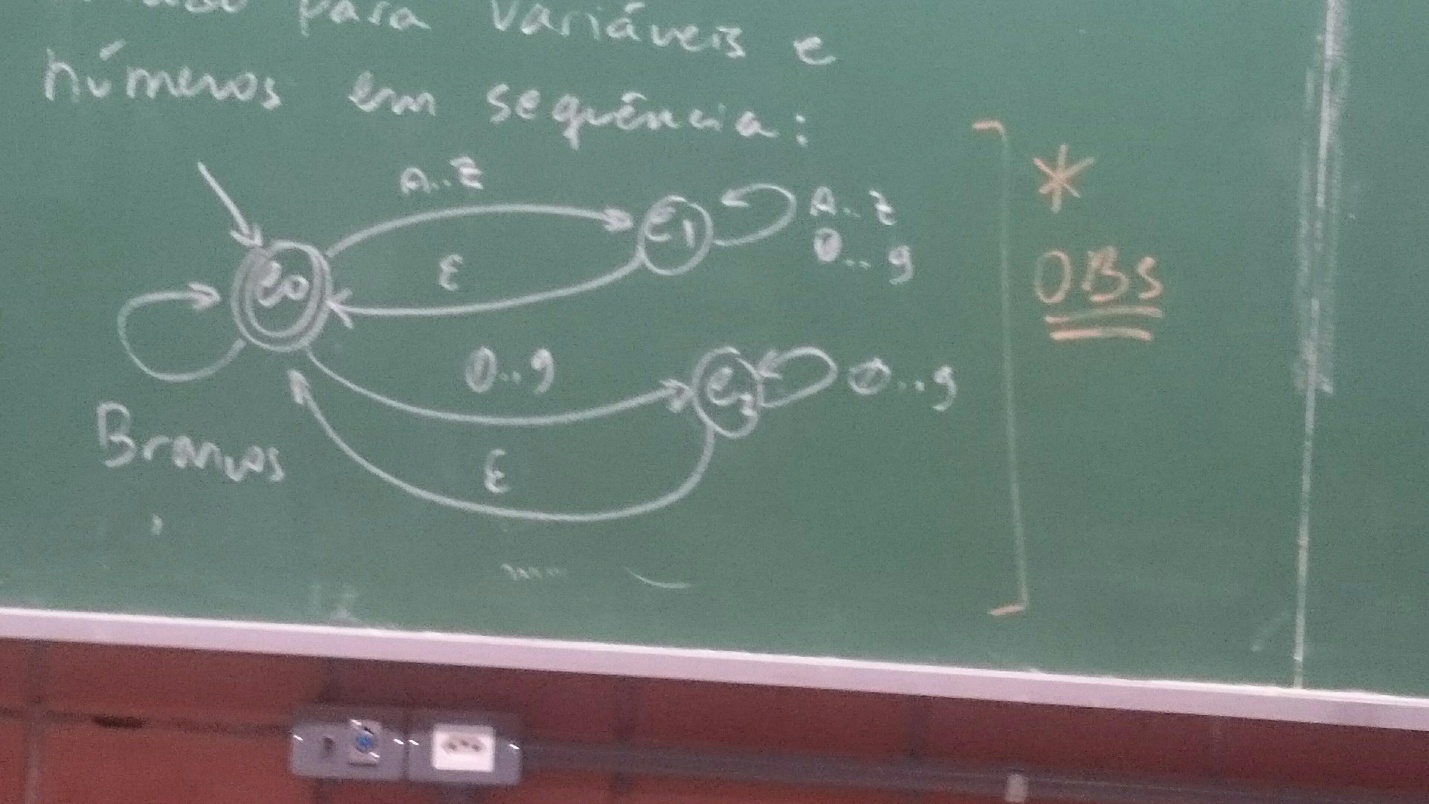
Transdutores são máquinas de estados que incluem ações em suas transições (\*)

Exemplo

Um possível AFD que reconhecem variáveis e números inteiros seriam:



Um AFN poderia ser criado para variáveis e número em sequência:



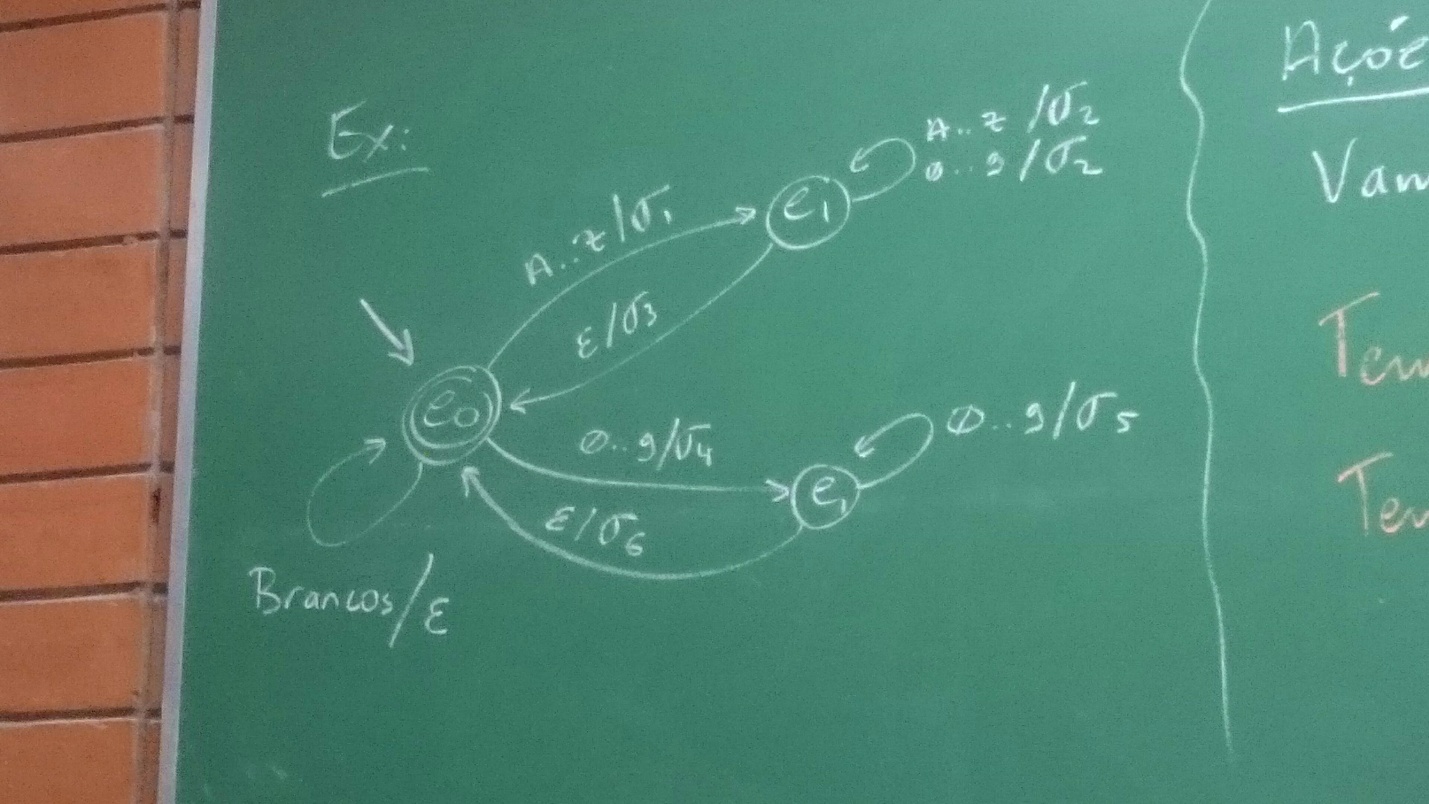
Para construir uma saída especial que use um “token” para cada caso, como no exemplo abaixo.

E: AB A AB 95 22

S: V(0)V(1)V(0)N(95)N(22)

Tabela de variáveis

Não seria possível com um AFD ou AFN. Para isso, usaremos um transdutor.

EX:

Ações:

Vamos considerar que:

Exemplo:

E: A2 20 2A2 AS

S: V(0)N(20)N(2)V(0)V(1)

Na tabela V[0] = A2, V[1] = AS

# Análise Léxica

Considere em se definir os “tokens” da linguagem. Para isso usamos transdutores.

## Transdutor

sendo:

: Transição

: Ação

Considere uma linguagem com os seguintes elementos:

* variáveis
* números
* palavras reservadas
  + if
  + else
  + then
  + goto
  + print
  + read
  + let
  + end
  + of
* Sinais
  + >
  + <
  + =
  + !
  + :=
  + (
  + )
  + % (comentário)
* outros

**Um exemplo de entrada possível seria:**

let A:=10: print A: A:= A+1 : end %ABC

**Saída**

P(1) V(0) := N(10): P(3)V(0): V(0) := V(0) +N(1): P(8)

**Palavras reservadas:**

0. if

1. let

2. then

3. print

4. of

5. else

6. goto

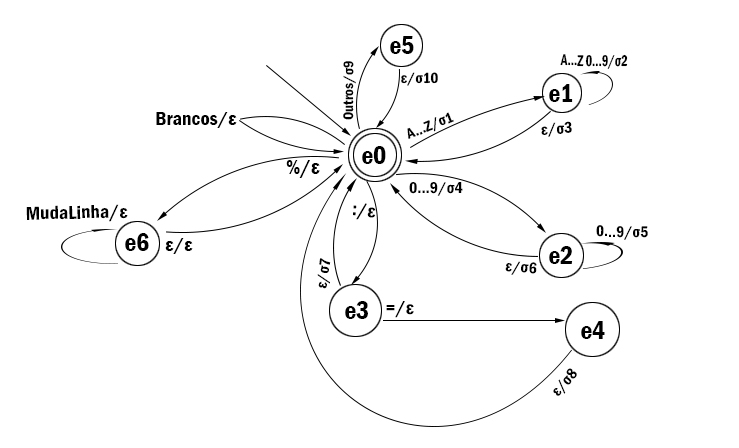
7. read

8. end

**Variáveis**

0….A

**Como montar isso?**



**Considerar que:**

TempS…. é um vetor char

TempN…. é um inteiro

TempC…. é um char

Há uma tabela de variáveis inicialmente vazia.

Assim:

….

TempS <- Simbolo

….

Anexa Simbolo em TempS

….

Finaliza TempS

**se** TempS est[a na tabela de palavras reservadas **então**

produz P([n° da posição na tabela])

**senão**

**se** TempS está na tabela de variáveis **então**

produz P([n° da posição na tabela]) na saída

**senão**

Cadastra variável na tabela

Produz P([n° da posição na tabela]) na saída

….

TempN <- símbolo - ‘0’

….

TempN <- TempN \* 10 + Simbolo ‘0’

….

Produz N([valor de TempN]) na saída

….

Produz “:”

….

Produz “;”

….

TempC <- símbolo

….

Imprime conteúdo de TempC na saída

**Exemplo:**

Entrada: AB:= AB + C : print AB

TempS -> AB TempC

Saída: V(0) := V(0) + V(1) : P(3) V(0)

Entrada: let X := X1 + X2

**Projeto 2: Construção do analisador léxico**

O programa irá operar da seguinte forma

Digite uma linha:

if(A>10) then B:=B+A %ABC

Saída:

P(2)(V(0) > N(10))P(4)V(1):=V(1)+V(0)

Palavras Reservadas:

[lista]

Variáveis:

[lista]

Prazo para entrega 13/09

# Gramáticas

São dispositivos geradores de palavras de uma linguagem

Definição

G = (V,T,P,S)

V = variáveis ou não-terminais

T = símbolos produzidos na saída ou não-terminais

P = regras de produção

S = Start (início)

Exemplo: dado que

Um possível gerador seria:

G = (V,T,P,S)

P

1. S -> A
2. A -> aA
3. A -> bA
4. A -> cA
5. A -> aC
6. C -> b

V = {S,A,C}

T = {a,b,c}

P = {S -> A, A -> aA, A -> bA, A -> cA, A -> aC, C->b}

S= S

Geração

Exemplo: para gerar

O conjunto de todas as palavras geradas por uma gramática é? Uma linguagem

Exercício

Dado que:

Pede-se:

Construir uma gramática que gere esta linguagem.

G = (V\_estadosnterminais,T\_simbolos,P,Start)

P =

G=( {S,A,B,C} , {1,0} , {...} , S)

## Exercício 2

Construir dado que

Construir a gramática

Conversão da gramatica para AF

|  |  |
| --- | --- |
| GR | AF |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Exemplo

Demonstração

|  |  |
| --- | --- |
| AF | GR |
|  |  |
|  |  |
|  |  |
|  |  |

# Compilador

(não esquecer as fotos sobre gramatica no tablet)

Um compilador é um tradutor de linguagem.

## Fases

Análise

Síntese

## Análise léxica

Entrada: linha de código/programa.

Saída: categorização dos elementos e tabelas variáveis.

## Análise Sintática

Entrada: saída de léxica.

Saída: sucesso ou erro.

Detalhes: é um reconhecedor AF/AP.

## Análise Semântica

Saída: código assembly não otimizado.

## Tipos de tradutor

Interpretador

O que é um interpretador:

Um tipo mais simples que checa e monta o código parcialmente para os comandos que ele interpreta internamente, exemplo SQL, PHP, SQL , Python.

## Montadores

O que é montador?

Programas mais simples exemplo, assembly, onde cada minemônico faz uma montagem, exemplo em assembly jmp = 0050 obs: assembly linguagem máquina assembler é o montador.

### Macro montador

O que é um macro montador

É um montador de macro, um comando que substitui outros comandos.

## Analises

Léxica, sintática e semântica

Léxica: pega a linha de código e cria a tipicação dos elementos (reservada, símbolo, tabelas de variáveis)

### Sintática

O que é analise sintática

Analise que permite ver se está correto a linha escrita, no caso erro de digitação no código. (i.e: sujeito, verbo direto, verbo indireto, com isso você constrói a análise da sentença gramatical)

### Semântica

O que é análise semântica

Obs: Autômato é um reconhecedor de máquina de estado (dispositivo denotacional), Gramática (dispositivo geradores),

Compilador: => Desenhar gramatica, desenhar os autômatos, optimizou os épsilon

Máquina de comando

Trabalho:

“digite expressão:”

Saídas: aceito/rejeitado;