Relatório LFA
Resolução dos exercícios feito em aula

Nome: Rosangela Miyeko Shigenari 92334
Thauany Moedano 92486
Kaio Henrique Dantas Maia 92420
Nickollas de Oliveira Aranha 77405

Aula 03/03

1.) Seja Gn = $\sum_{k=1}^{n}$ (k.k!)

Faça a demonstração por indução que a fórmula fechada de Gn é Gn = (n+1)! - 1

Resposta:

Base: n = 1.
G0 = (1+1)! -1 = 2 - 1 = 1.

Hipótese: K = n - 1
Gn-1 = (n+1-1)! - 1 = n! -1

Passo indutivo:
Gn = $\sum_{k=1}^{n-1}$ +(n.n!)
= (n! - 1) + (n.n!)
= n!(1+n) - 1
= (n+1)! - 1

2.) Prove por indução que a soma dos cubos de três inteiros positivos consecutivos é sempre divisível por 9.

Resposta:

Base: n = 0
$2^3 + 1^3 + 0^3 / 9 = 1$

Hipótese: k = n
$n^3 + (n-1)^3 + (n-2)^3$ é divisível por 9.

Passo indutivo:
$(n+1)^3 + n^3 + (n-1)^3 + [(n-2)^3 - (n-2)^3]$
= $[n^3 + (n-1)^3 + (n-2)^3] + [(n-1)^3 - (n-2)^3]$

A primeira parcela da equação é divisível por 9 pela hipótese indutiva. Basta mostrar que a segunda parcela também é pois pela teoria matemática um número divisível por 9 somado a número divisível por 9 resulta em um número também divisível por 9.

$(n+1)^3 + (n-2)^3$

$n^3 + 3n^2 + 3n + 1 - n^3 - 2n^2 - 4n^2 + 12n - 8$

$9n^2 + 9n + 9$ que é um número divisível por 9.

Aula 10/03

1.) {w | w tem a forma "x01y", onde x{0,1}*}

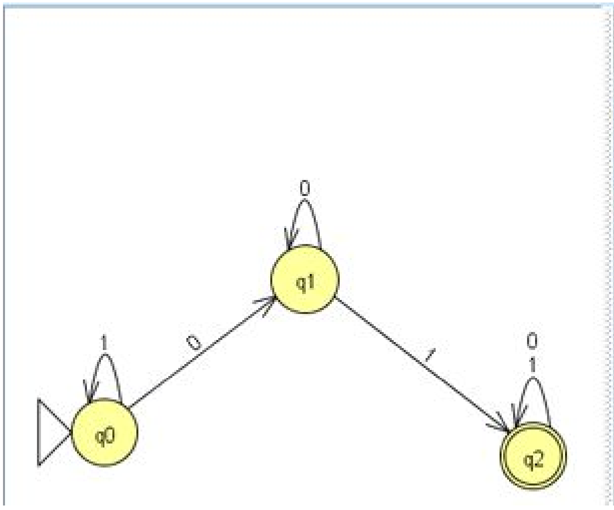


Table Text Size

Input	Result
11111	Reject
0000	Reject
110100	Accept
1110	Reject
0001	Accept

2.) {w | w tem um número par de "0"s}

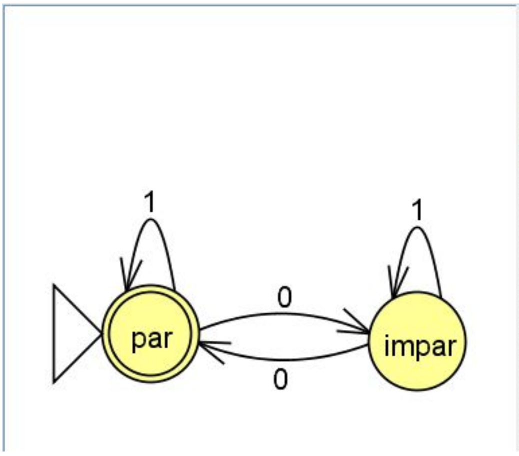


Table Text Size

Input	Result
11111	Accept
00110	Reject
1101110010	Accept
001100	Accept

3. {w | w tem ao mesmo tempo um número par de "0"s e "1"s}

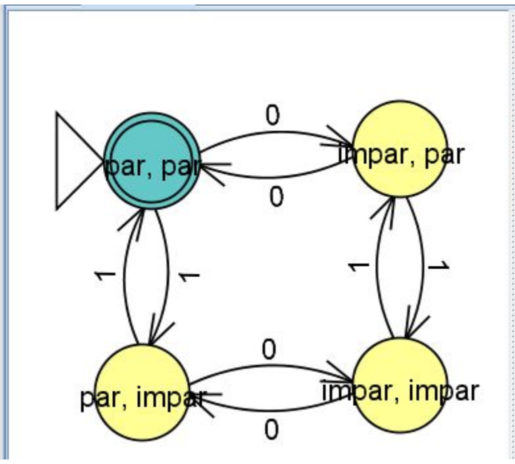


Table Text Size

Input	Result
11000	Reject
0011	Accept
000111	Reject
0101010101	Reject
1101	Reject
1111	Accept

4. {w | w termina em "00"}

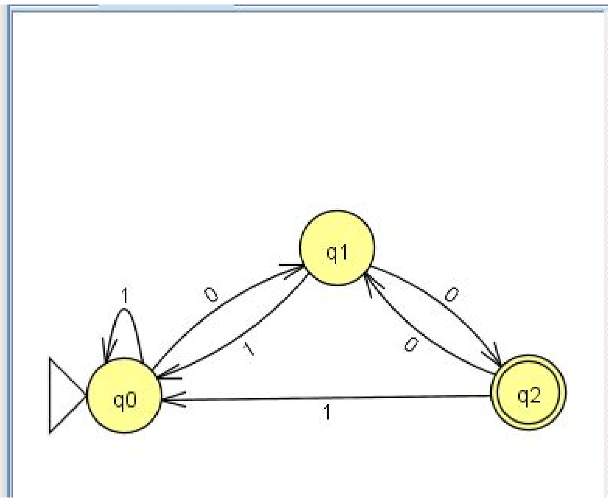


Table Text Size

Input	Result
111	Reject
01100	Accept
11010	Reject
1100	Accept

5. {w | w tem três "0"s consecutivos}

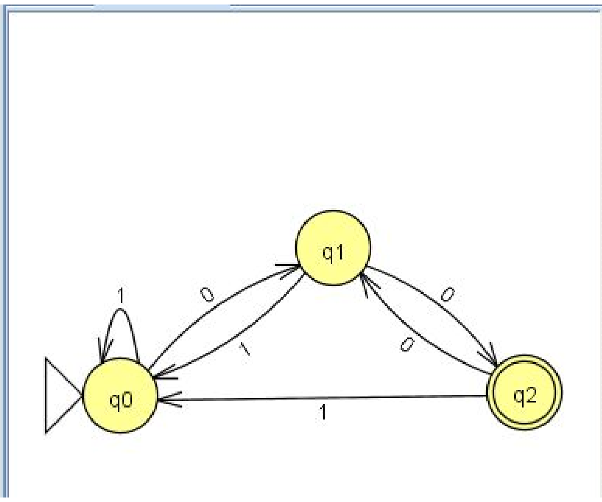


Table Text Size

Input	Result
111	Reject
01100	Accept
11010	Reject
1100	Accept

6. {w | w tem "011" como substring}

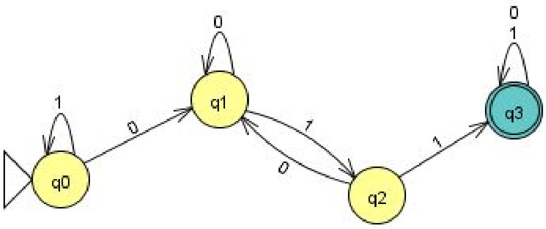


Table Text Size

Input	Result
111	Reject
011101	Accept
110101011110	Accept
00000	Reject
0	Reject
1	Reject
111100000	Reject
101010011	Accept

7. {w | w tem número de "0"s divisível por 5 e de "1"s divisível por 3}

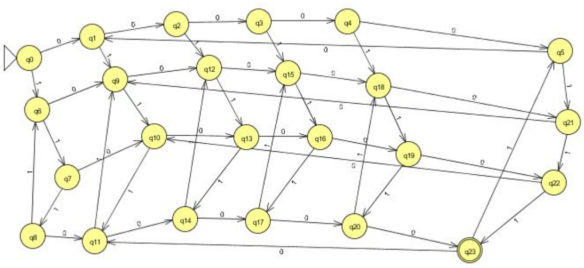


Table Text Size

Input	Result
00011111	Reject
00000111	Accept
011101110111101110	Accept
01010101	Reject
110001110	Reject
100010101	Reject

Aula 15/03

Desenhe linguagens que aceitem as seguintes linguagens sobre o alfabeto {1,0}
1- Todos os strings que contêm exatamente 4 0’s.

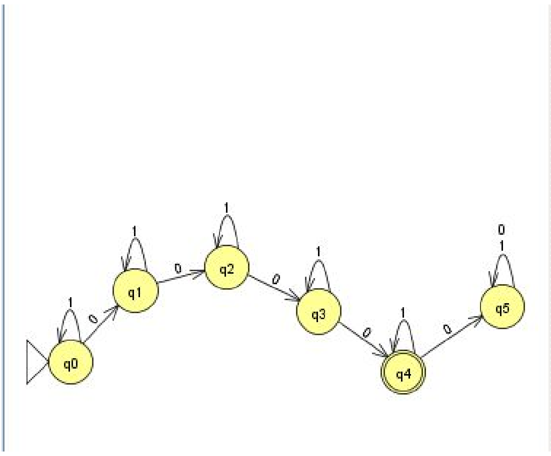


Table Text Size

Input	Result
10100110	Accept
1110010110011	Reject
000011111	Accept
1010101010	Reject

2- Todos os strings que terminam em “1101”.

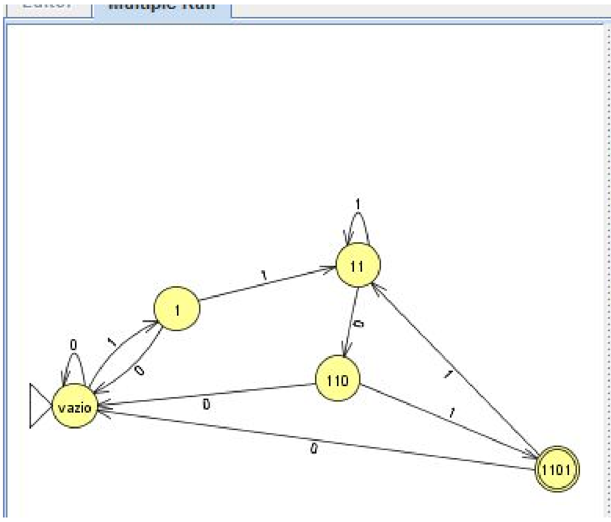
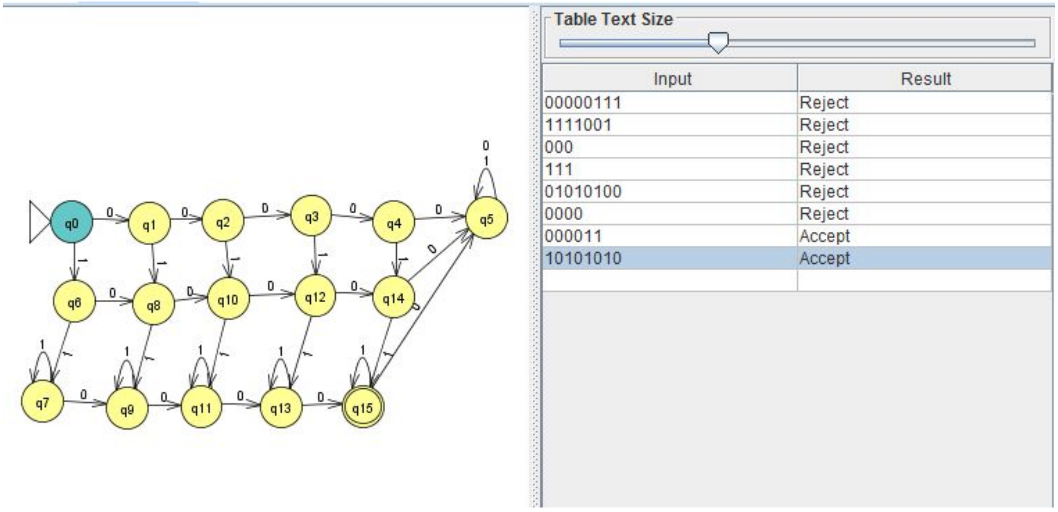


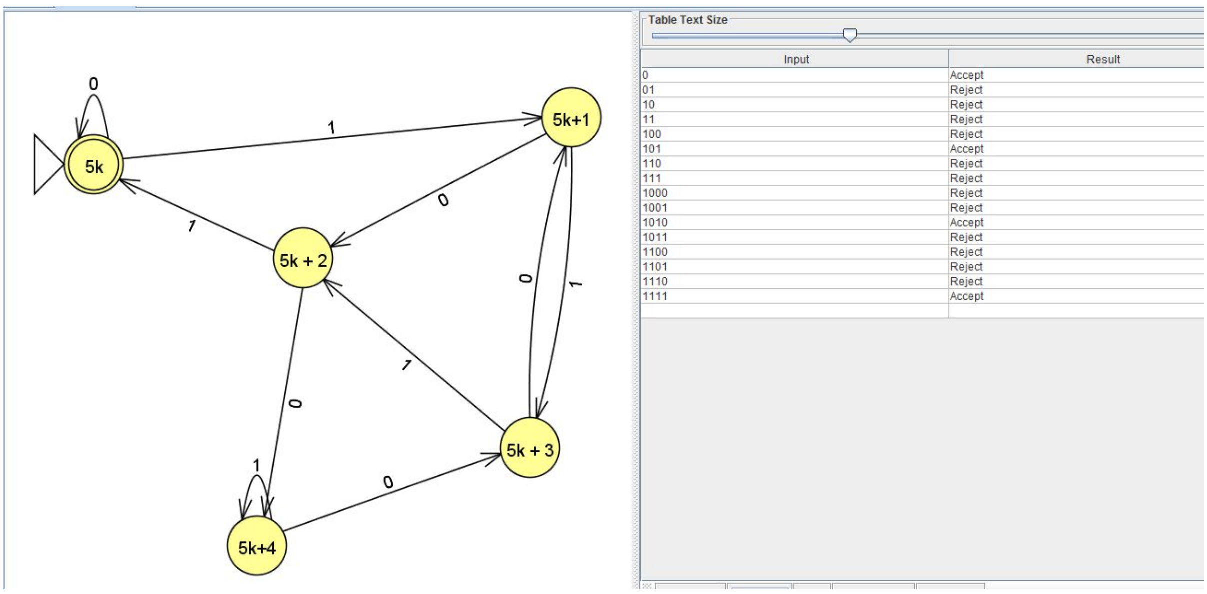
Table Text Size

Input	Result
010101101	Accept
01011100	Reject
1101110	Reject
0	Reject
1	Reject
10	Reject
10010011101	Accept

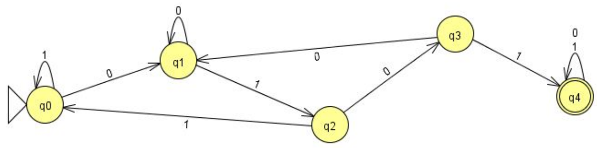
3- Todos os strings que contêm exatamente 4 0’s e pelo menos 2 1’s.



4- Todos os strings cujo valor binário é divisível por 5.

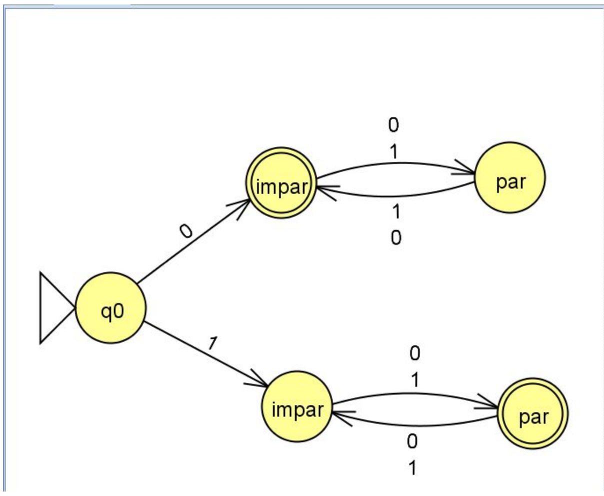


5-Todos os strings que contêm o substring “0101”.



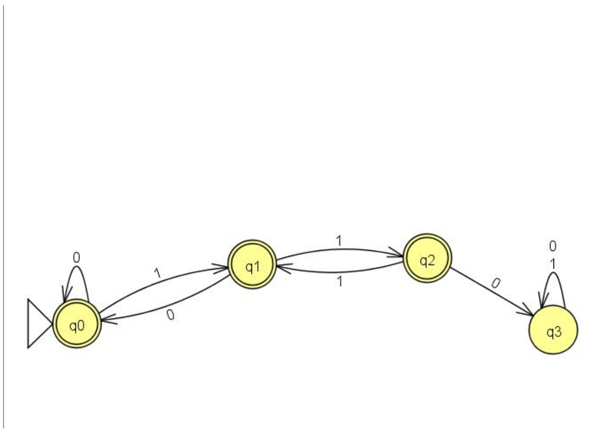
Input	Result
0101	Accept
1100	Reject
000010000	Reject
01010101010101	Accept
10101010000	Accept

6- Todos os strings que começam com 0 e têm comprimento impar ou começam com 1 e têm comprimento par.



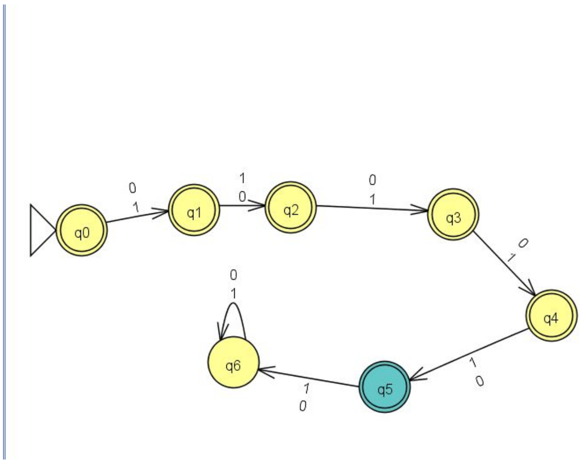
Input	Result
0101	Reject
1100	Accept
000010000	Accept
01010101010101	Reject
10101010000	Reject
110	Reject
0111	Reject
11	Accept
000	Accept

7- Todos os strings que não contêm o string 110.



Input	Result
0101	Accept
1100	Reject
000010000	Accept
01010101010101	Accept
10101010000	Accept
110	Reject
0111	Accept
11	Accept
000	Accept

8- Todos os strings com comprimento no máximo igual a 5.



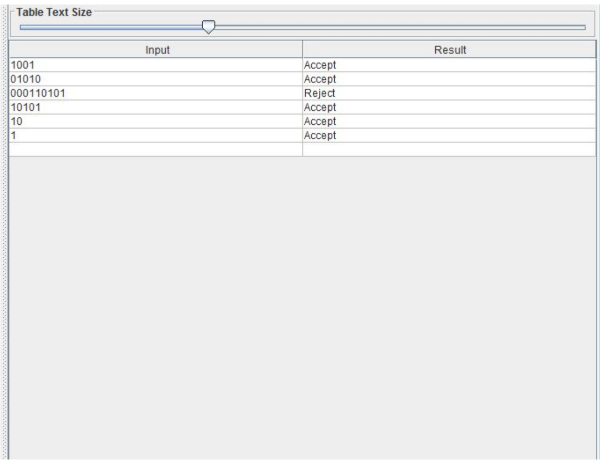
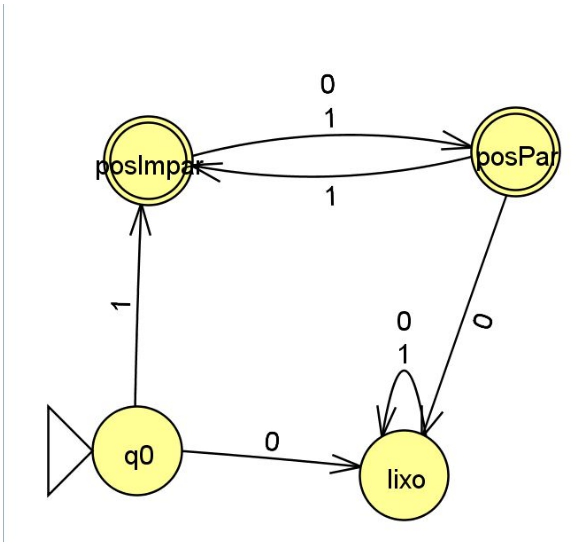


Table Text Size

Input	Result
1001	Accept
01010	Accept
000110101	Reject
10101	Accept
10	Accept
1	Accept

9- Todos os strings que têm 1 em posições impares



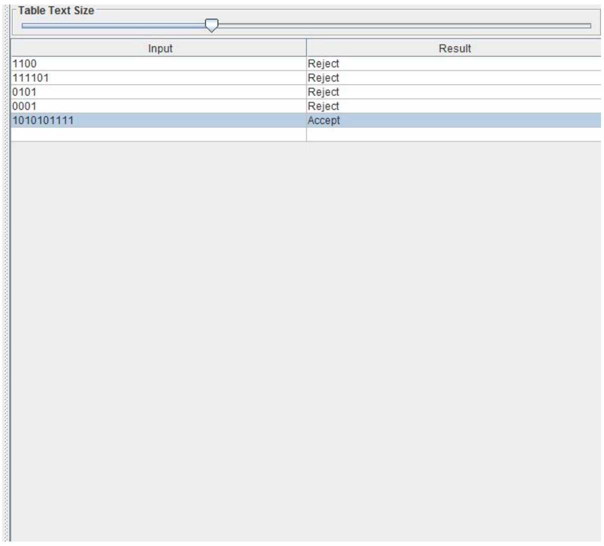


Table Text Size

Input	Result
1100	Reject
111101	Reject
0101	Reject
0001	Reject
1010101111	Accept

Aula 17/03

1) Construir DFA para as seguintes linguagens

a){w ∈ {0,1}* | cada 0 de w é imediatamente seguida de no minimo dois 1’s}

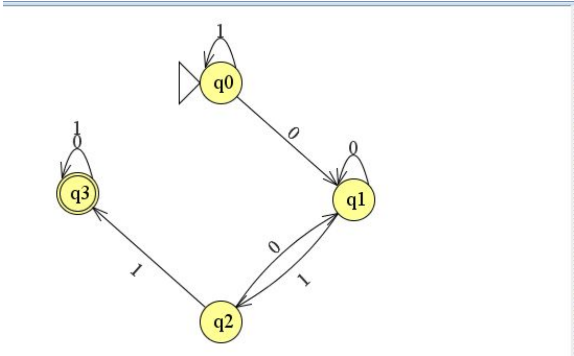


Table Text Size

Input	Result
0 0 0 1 1	Accept
1 0 1 1 0	Accept
1 0 0	Reject
1 1 1	Reject
1 0 0 0 1	Reject
1 1 0 0 1 1 1 1	Accept
λ	Reject

b){w ∈ {0,1}* | w não contém nem 000 nem 111.

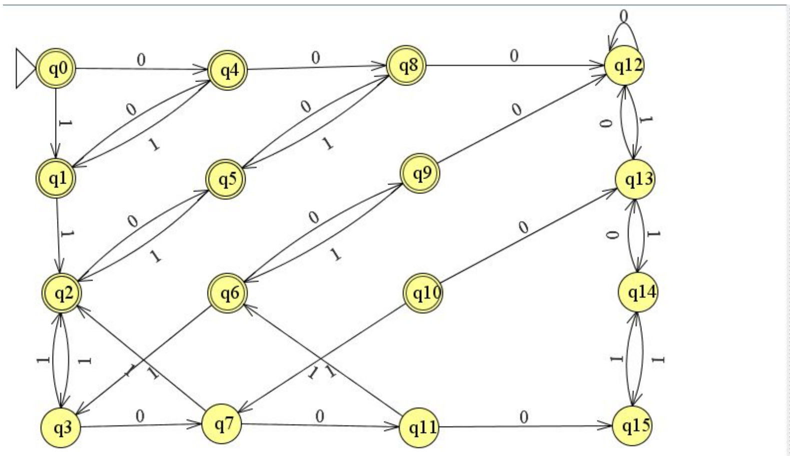


Table Text Size

Input	Result
0 0 0 1 1	Reject
1 0 1 1 1 0 0	Reject
1 0 0	Accept
1 1 1 0 0 0 1 0	Reject
1 0 0 0 1 0 1	Reject
1 1 0 0 1 1	Accept
1 0	Accept
1 0 1 1 1 1	Accept
λ	Accept

c){w ∈ {a,b}* | w começa com a e tem tamanho par.

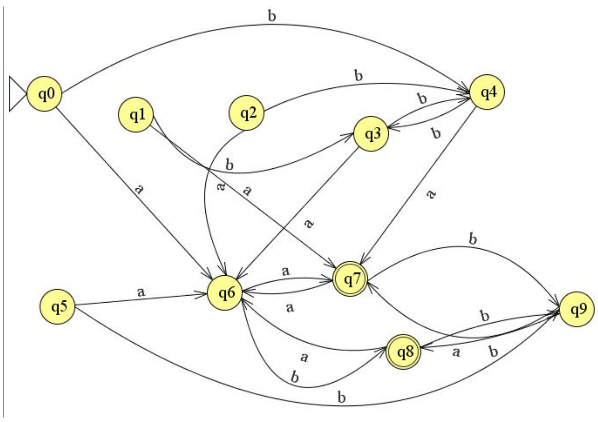
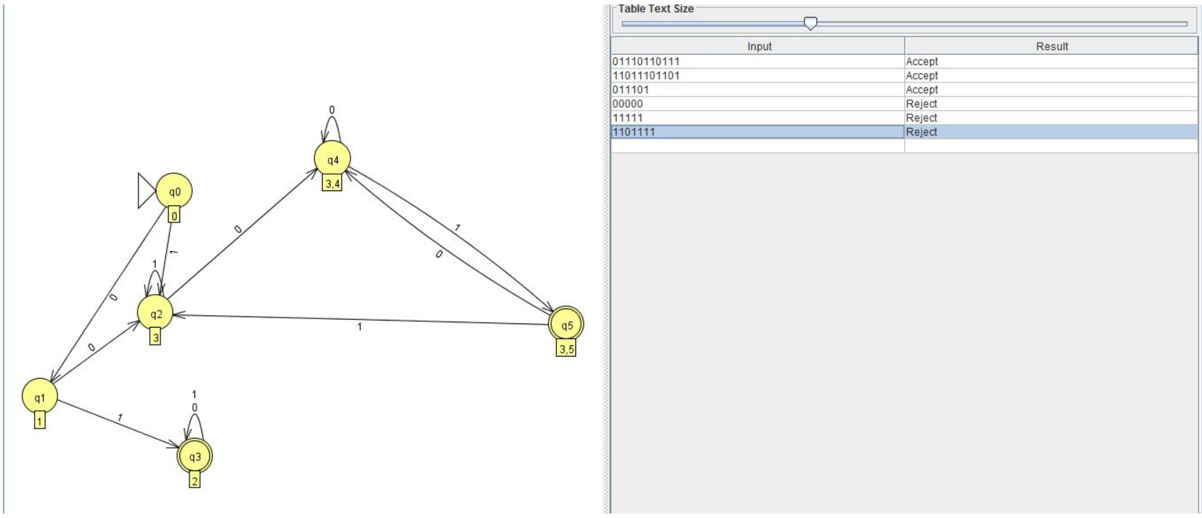


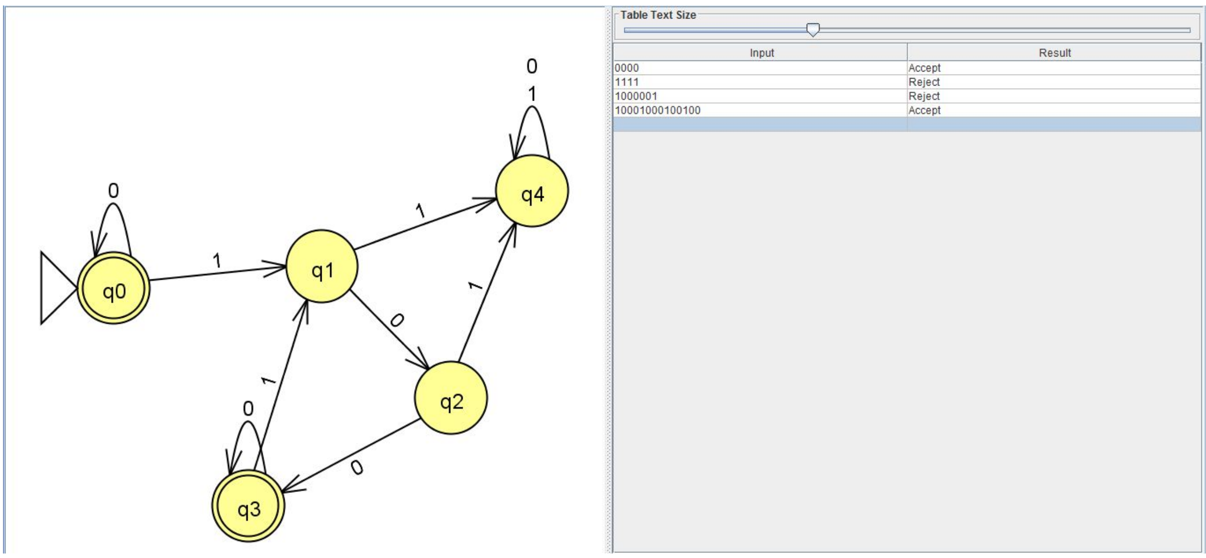
Table Text Size

Input	Result
a a	Accept
a b a b a b	Accept
b b b b	Reject
a b b	Reject
b a b a a	Reject
b a a	Reject
a b b a a b a	Reject
λ	Reject
b	Reject
a	Reject

Load Inputs | Run Inputs | Clear | Enter λ | View Trace d){w ∈ {0,1}* | w começa, ou termina, ou ambos com 01.



e){w ∈ {0,1}* | w apresenta cada 1 seguido imediatamente de dois 0's.



2) Implementar um programa na linguagem capaz de ler dois arquivos, sendo que o primeiro contém a descrição de um DFA e o segundo uma string para o reconhecimento.

Linha 1: Estados
Linha 2: Alfabeto
Linha 3:Estado inicial
Linha 4: Estados finais

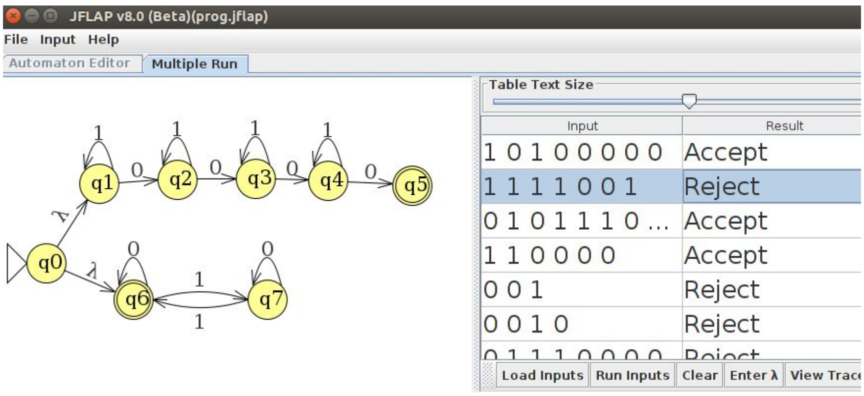
Linha 5: Em frente tabela de transição

```cpp
#include <iostream>
#include <string>
#include <algorithm>
#include <vector>

using namespace std;


void transforma(string& input, const string& oldInput, const std::string& nInput){
    size_t pos = 0;

    while(true){
        input.replace(pos, oldInput.length(), nInput);
        pos += nInput.length();

    if((pos = input.find(oldInput, pos)) != std::string::npos)
        break;
    }
}

int delta(int ini, vector< vector<int> >& tabela, string valor){
    int first = 0, aux, res = ini;

    for, (i = 0, i < valor.size(), i++){
        aux = (int) valor[i] - 48;
        result = tabela[ini+1][aux+1];
        ini = resut;
    }

    return result;
}
```

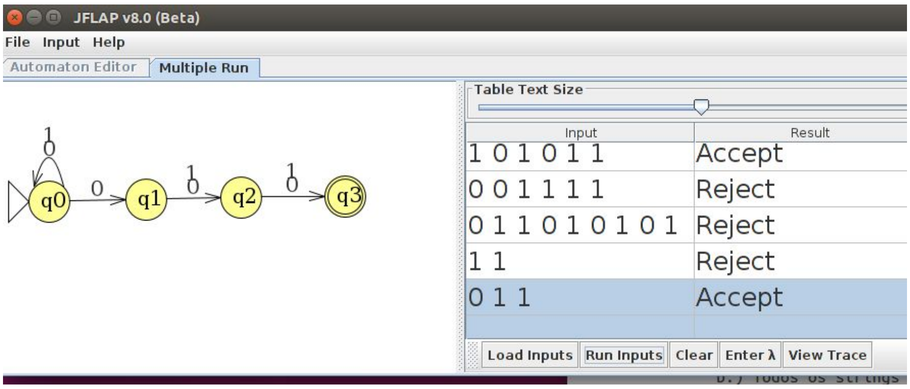
Aula 22/03

1.) Construir NFA's para as seguintes linguagens regulares:

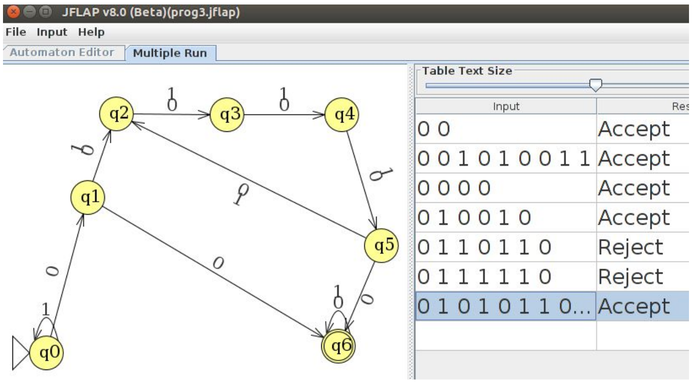
a.) Todos os strings que contem exatamente 4 0's ou um n par de 1's



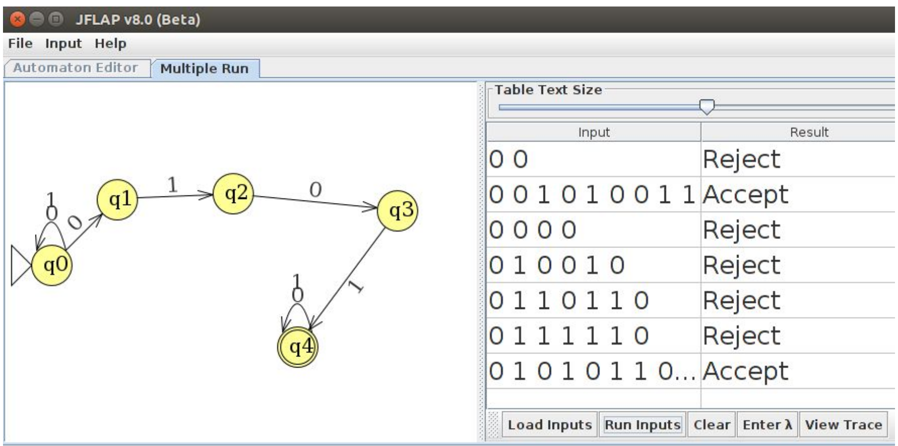
b.) Todos os strings em que o terceiro símbolo a partir da direita é 0



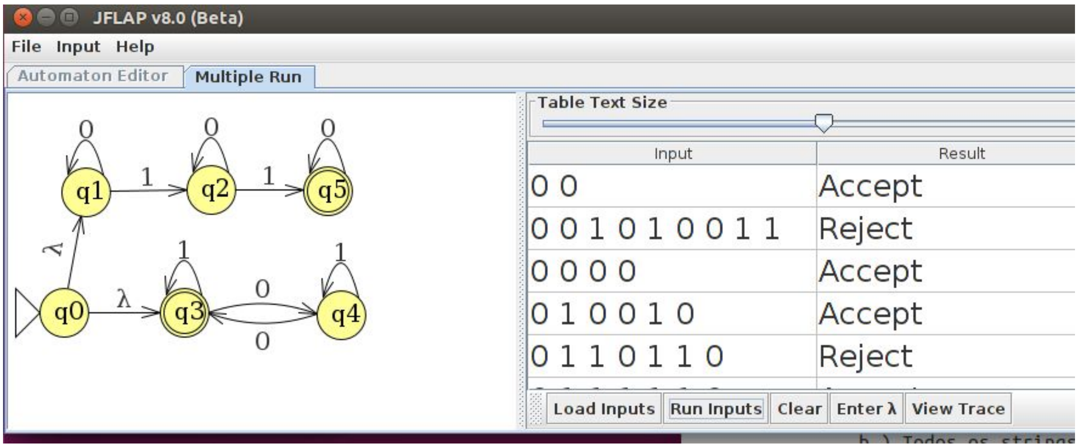
c.) Todos os strings em que ocorrem 2 0's separados por um string de comprimento 4i para algum i >= 0



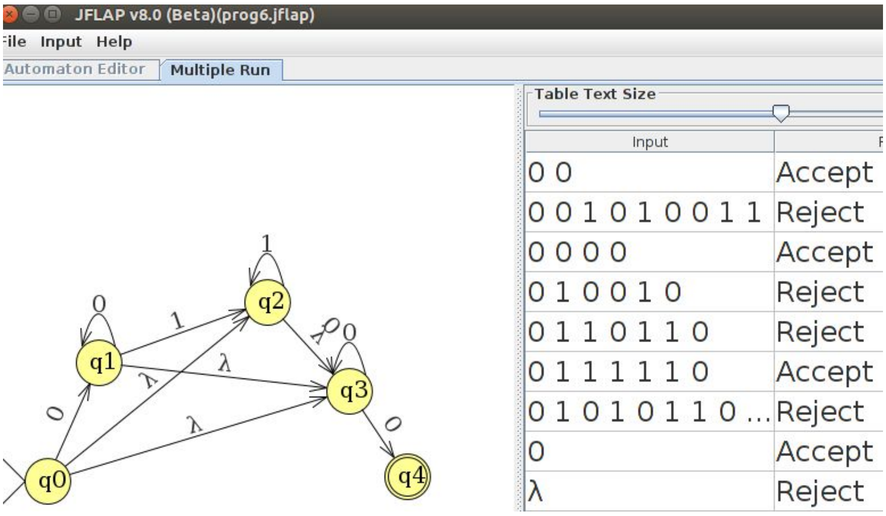
d.) Todos os strings que contem o substring 0101



e.) Todos os strings que contem um num par de 0`s ou exatamente dois 1’s



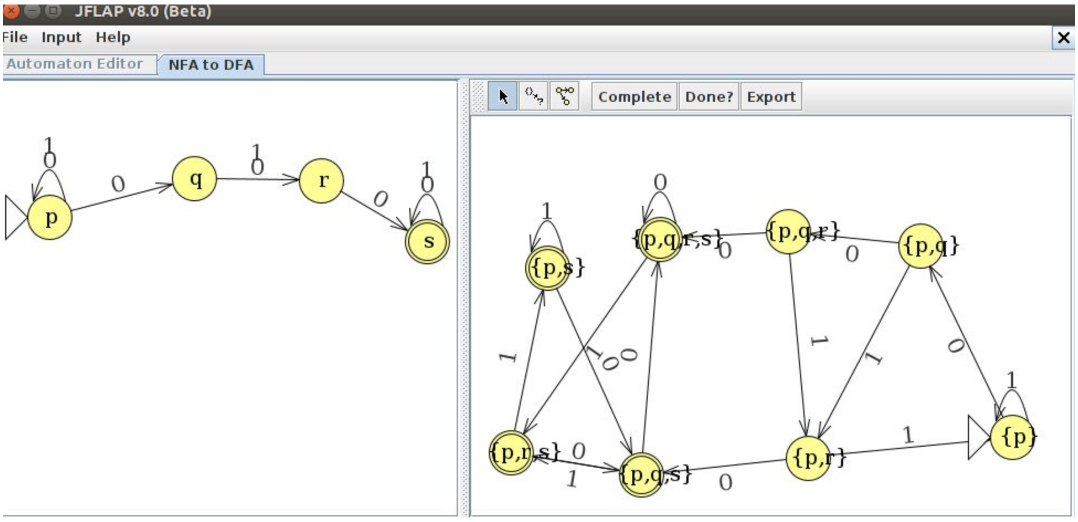
f.) 0* 1* 0* 0



2.) Faça dois exemplos de transição para cada NFA produzido na questão 1
Resposta: As transições estão no próprio exercício 1

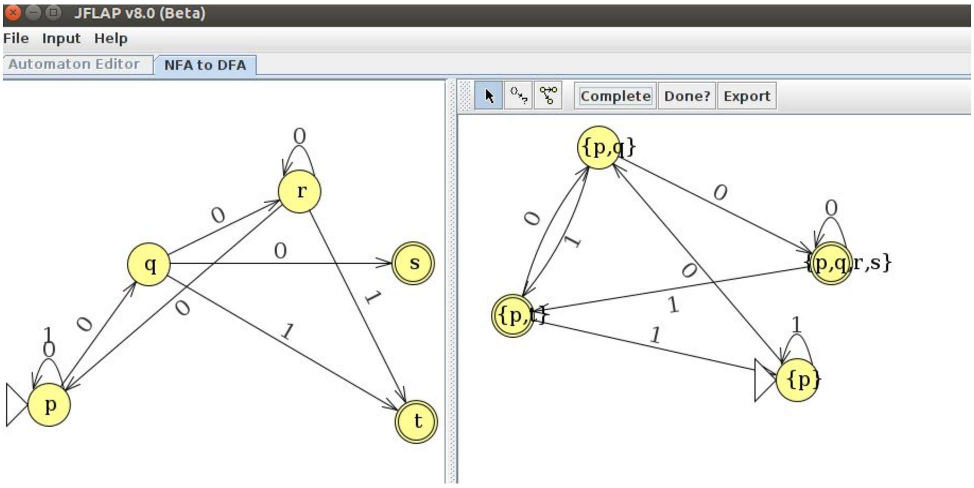
Aula 24/03

1.) Mostre que a seguinte linguagem eh regular: L = {conjunto de strings que contêm igual número de ocorrências de 01 e 10}



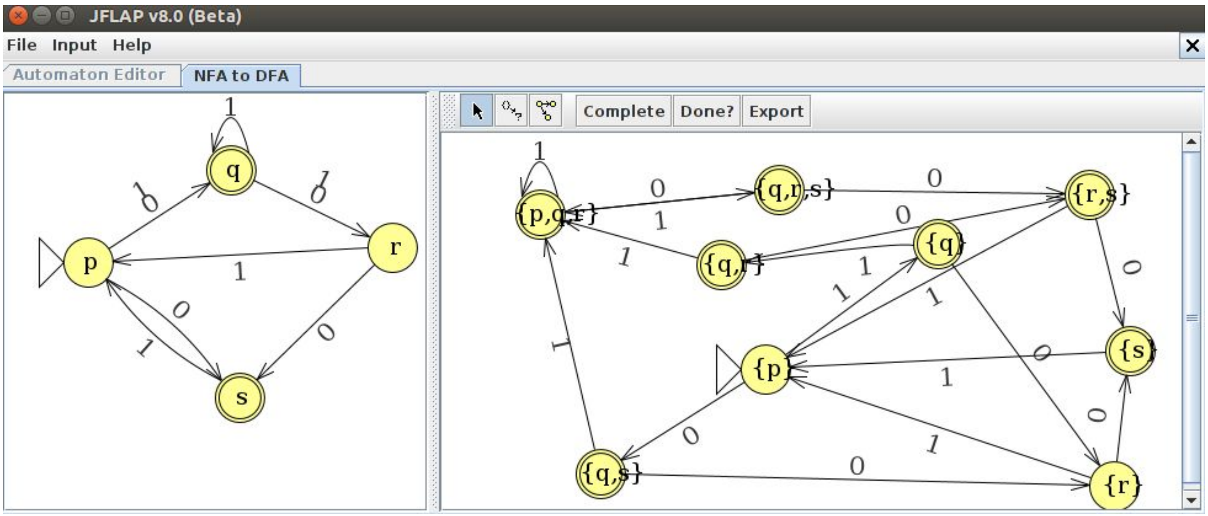
2.) Converta o seguinte NFA em DFA

	0	1
->p	**{p,q}**	**{p}**
q	**{r}**	**{r}**
r	**{s}**	λ
***s**	**{s}**	**{s}**



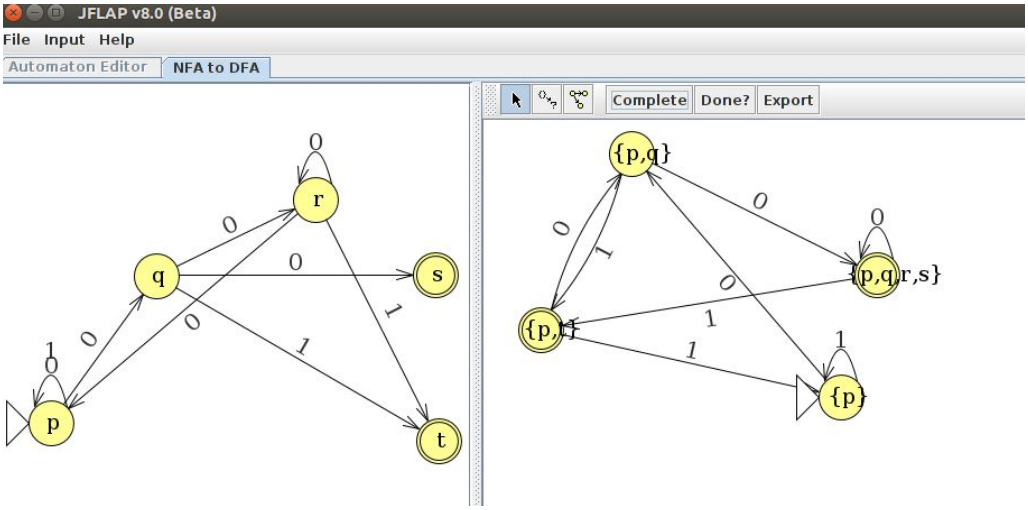
3.) Converta o seguinte NFA em DFA

	0	1
->p	{q,s}	{q}
*q	{r}	{q,r}
*r	{s}	{p}
*s	λ	{p}

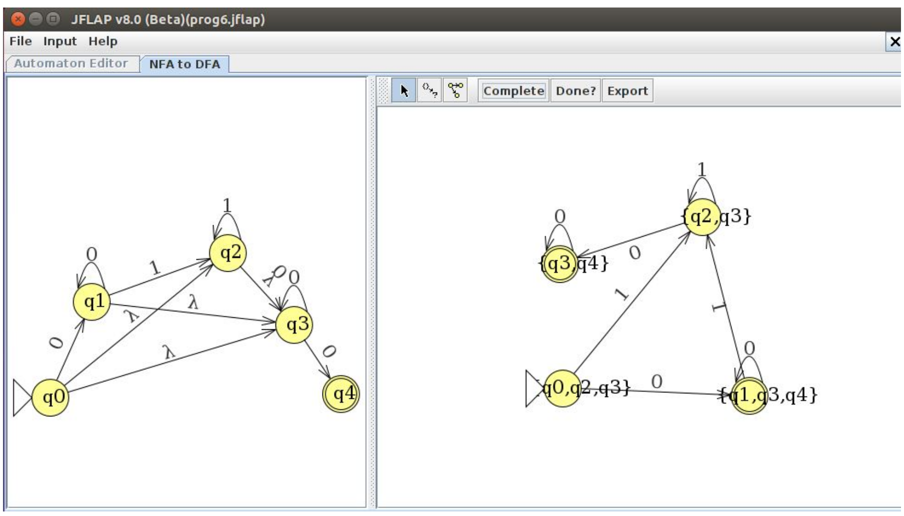


4.) Converta o seguinte NFA em DFA

	0	1
->p	**{p,q}**	**{p}**
q	**{r,s}**	**{t}**
r	**{p,r}**	**{t}**
***s**	λ	λ
***t**	λ	λ



5.) Converta o seguinte NFA da Linguagem 0*1*0*0 em DFA



Aula 29/03

1.) Considere o NFA

	λ	a	b	c
->p	λ	**{p}**	**{q}**	**{r}**
q	**{p}**	**{q}**	**{r}**	λ
***r**	**{q}**	**{r}**	λ	**{p}**

a.) Calcule o ε-fechamento de cada estado

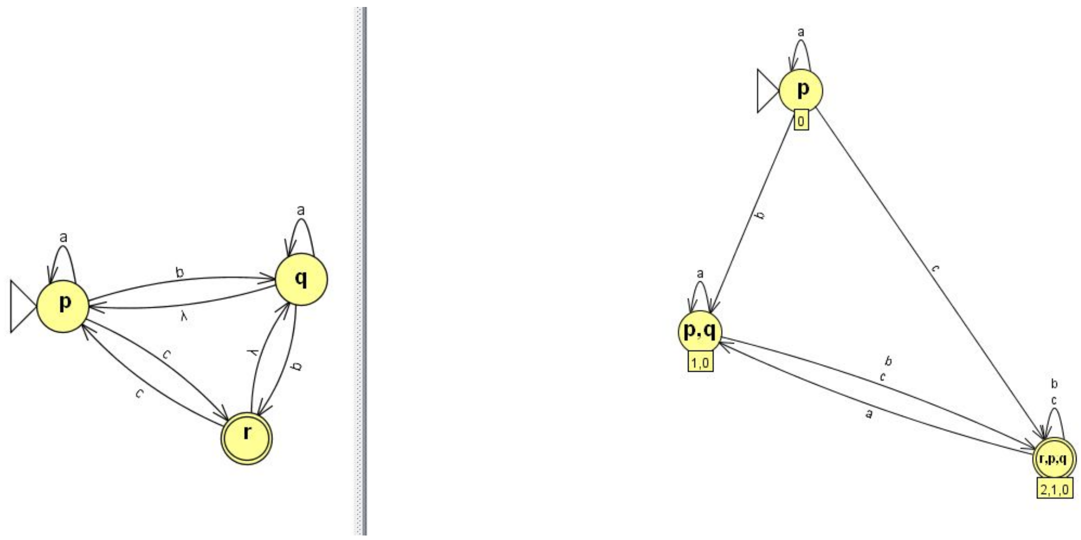
Resposta:

p: {p}

q: {q, p}

r:{r, p,q}

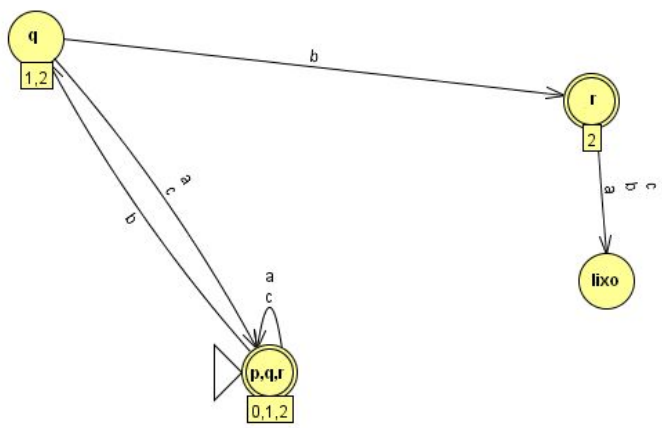
b.) Converta o autômato para DFA



2.) Repita o exercício 01 para

	λ	**a**	**b**	**c**
->p	**{q,r}**	λ	**{q}**	**{r}**
q	λ	**{p}**	**{r}**	**{p,q}**
***r**	λ	λ	λ	λ

Resposta:
p: {p,q,r}
q: {q}
r: {r}



3.) Prove que a linguagem é regular: Conjunto de todas strings tais que cada bloco de 5 símbolos consecutivos contem pelo menos dois 0's.

Resposta: Podemos provar que uma linguagem é regular a partir do Lema do Bombeamento: que segue as seguintes propriedades

- u = xyz (x é um prefixo, z é um sufixo)
- |y | ≥ 1 (a parte do meio y é não vazia)
- |xy| ≤ p (bomb. nos p primeiros simbolos)
- xy^iz ∈ L $\forall i \geq 1$

Tomando que p = 7 e S representa os símbolos

<u>S</u>	<u>S</u>	<u>S</u>	<u>S</u>	<u>S</u>	<u>0</u>	<u>0</u>

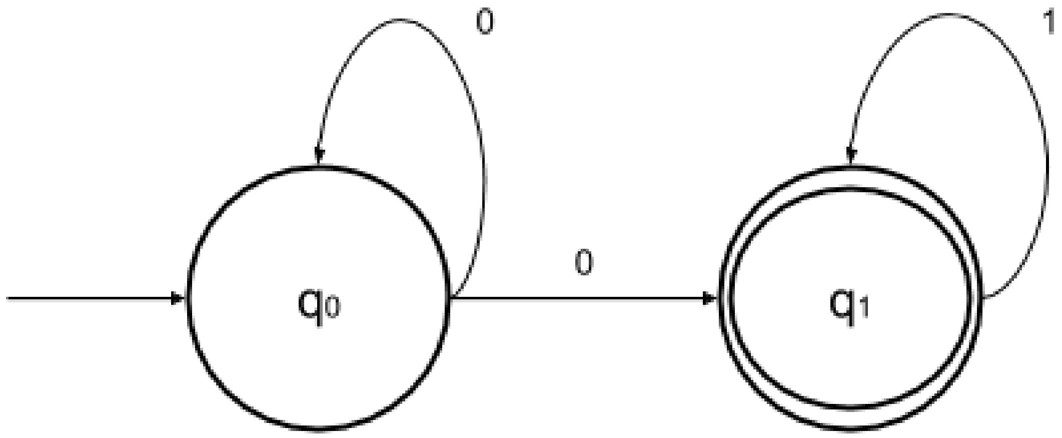
x y z

Para o caso base representado acima, temos:

- u = SSSSS00
- |y| = 1
- |xy| ≤ 7
- xy^1z ∈ L $\forall i \geq 1$

A RE é $(S^5\ 0^n)$* para n≥2
Podemos escrever que o primeiro bloco de símbolos será o x da nossa string, o y será o primeiro bloco de 0's e o z como sendo o último 0.
Assim, para $\forall i \geq 1$ garanto que xy^iz pertence à minha linguagem e a string terá no mínimo 2 zeros seguidos de um bloco com 5 símbolos.

4.) Considere o autômato M = ({q0, q1}, {0,1}, L, q0, {q0, q1}) onde a função L é especificada pelo diagrama

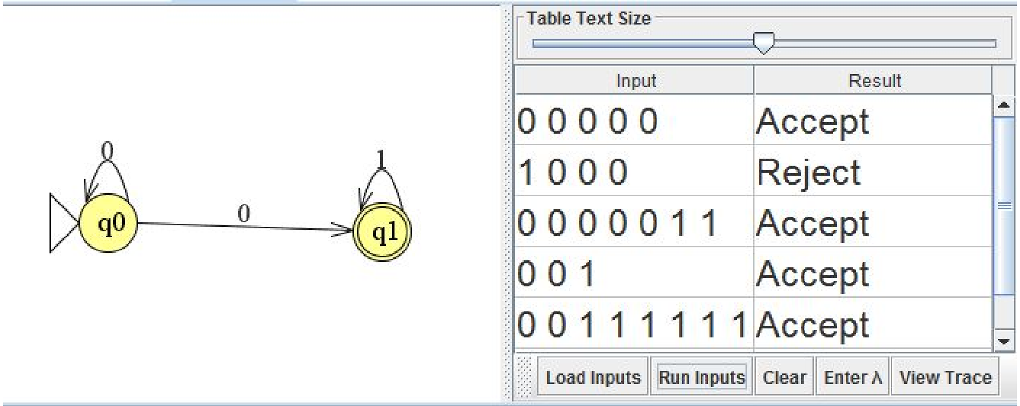


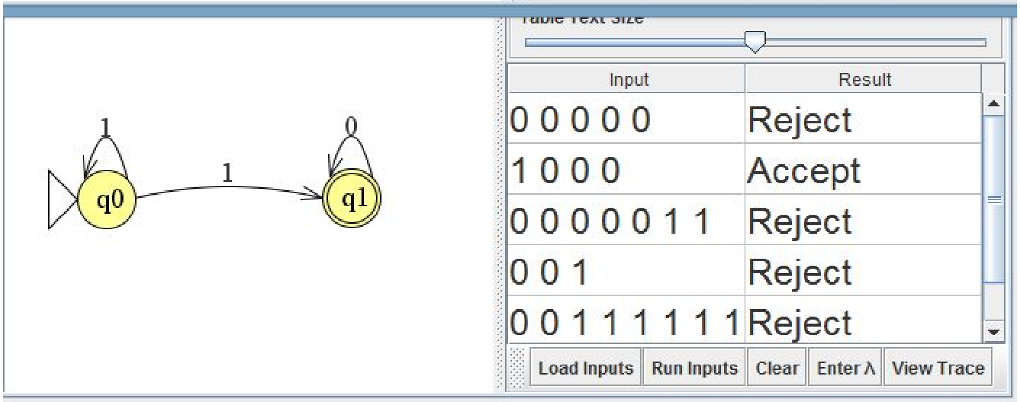
a.) Qual a linguagem reconhecida por M ?
Resposta: Todas as strings que começam com 0 e terminam com 1
Linguagem que reconhece a RE 01*

b.) Considere M' = ({q0, q1}, {0,1}, δ, q0, ε} M' é automato complementar de M?
Resposta: Não, para que seja um autômato complementar, as palavras que não são aceitas por M terá que ser aceita por M', o que não ocorre já que o estado de aceitação é ε.

c.) Construa o automato complementar de M

Resposta: O primeiro é o M e o segundo o M’ (complementar)





Aula 31/03

Crie expressões regulares ∑ = {0,1}

1.) {w | w contêm um único 1}

Resposta:

L = (0)*1(0)*

2.) {w | w contêm pelo menos um 1}

Resposta:

L = 0*(1 + 1*)(0+1)*

3.) {w | w contêm a string 001 como substring}

Resposta:

L = (1*+0*)001(1*+0*)

4.) {w | |w| e par}

Resposta:

L = ((0+1)(0+1))*

5.) {w | |w| e múltiplo de 3}
Resposta:
L = ((0+1)(0+1)(0+1))*

6.) {w | 6 simbolo direita/esquerda e 1}
Resposta:
L = (0+1)*1(0+1)(0+1)(0+1)(0+1)(0+1)

7.) Descreva em português o conjunto denotado por cada uma das expressões regulares:

a.) 1*0
Resposta: Todas as strings que possuem apenas 1's e terminam com zero

b.) 1*0(0)*
Resposta: Todas as strings que tem uma sequencia com qualquer quantidade de 1's seguida de uma sequencia de zeros que contenha pelo menos um zero.

c.) 111 + 001
Resposta: A string 111 ou 001

d.) (1 + 00)*
Resposta: Qualquer string que tem quantidade par de zeros com dois zeros consecutivos

e.) (0(0)*1)*
Resposta: Todas as strings que começam com zero, terminam com 1 e não aceitam 1's consecutivos

f.) (0+1)(0+1)*00
Resposta: Todas as strings de tamanho maior ou igual a 3 e terminam com 00.

Aula 05/04

1.) Conjunto de strings sobre o alfabeto {a,b,c} que contem pelo menos um a e pelo menos um b.
Resposta:
L = (a+b+c)*(ab)(a+b+c)* + (a+b+c)*(ba)(a+b+c)* + (ab)(a+b+c)* + (a+b+c)*(ab) + (ba)(a+b+c)* + (a+b+c)*(ba) + b(a+b+c)*a + a(a+b+c)*b

2.) Conjunto de todos os strings de 0’s e 1’s tais que todo par de 0’s adjacentes aparece antes de qualquer par de 1`d adjacentes

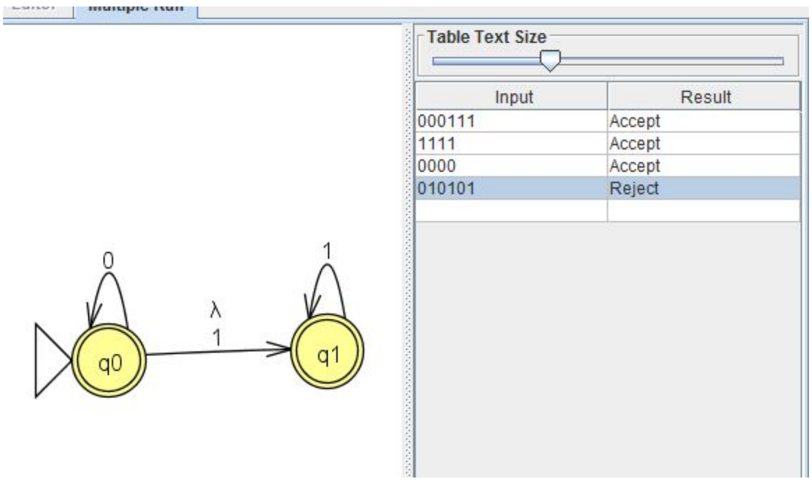
(0+10*)(e+1) - não há par de um
(1+01*)(e+0) - não há par de zeros
Resposta:
L = (0+10*)(e+1)(1+01*)(e+0)

3.) O conjunto de todos strings que não contem o substring 101
Resposta:
L = (1+e)(0+100+11)*(1+e)

4.) O conjunto de todos os strings que tem no máximo um par de 0’s consecutivos ou um par de 1’s consecutivos.
Resposta:
L = (1+e)(01)*(00)(10)*(1+e) + (0+e)(10)*(11)(01)*(0+e) + (1+e)(01)*(0+e) + (0+e)(10)*(1+e)

5.) Dar um ε-NFA equivalente as expressões regulares

a.) 0*1*



b.) 00*11*

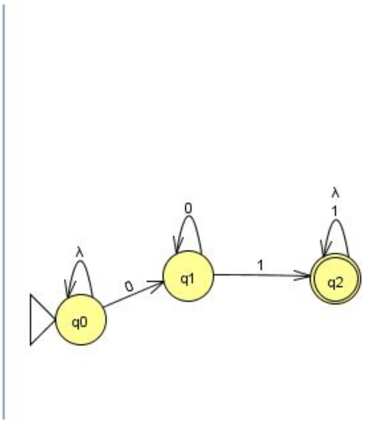


Table Text Size

Input	Result
000111	Accept
1111	Reject
0000	Reject
010101	Reject
01	Accept
10	Reject

c.) (00)*(11)*

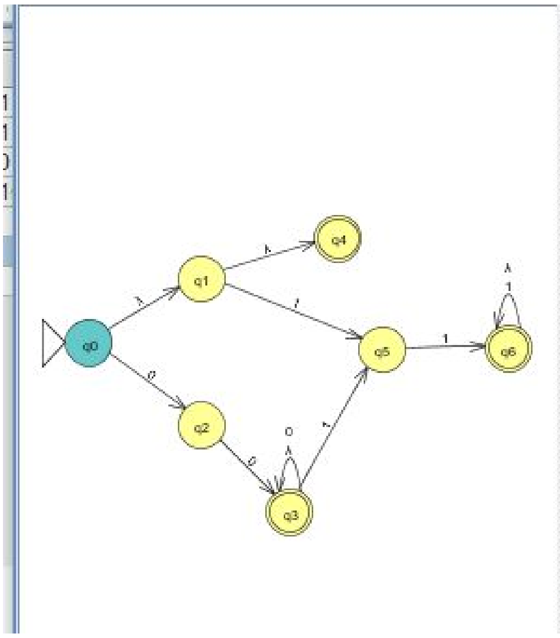


Table Text Size

Input	Result
000111	Accept
1111	Accept
0000	Accept
010101	Reject
01	Reject
10	Reject
0011	Accept
011	Reject
001	Reject
	Accept

d.) (0+1)(0+1)*00

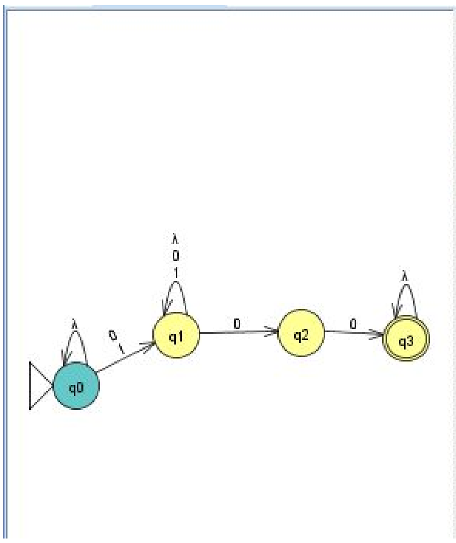
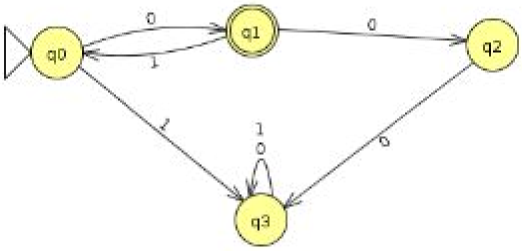


Table Text Size

Input	Result
10101011	Reject
010101100	Accept
1011100100	Accept
0101110011011	Reject
00	Reject

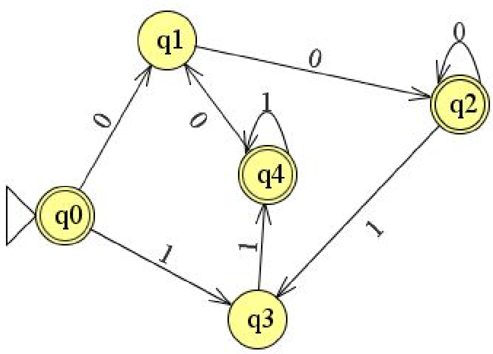
6.) Determine a RE da linguagem aceita pelo autômato abaixo



Resposta: A linguagem aceita é o ciclo 01, que pode ou não existir seguido de 0, escrevendo a RE temos (01)*0.

7.) Construir o automato correspondente a RE ((00)*(11)*)*, utilizando o algoritmo que transforma RE em autômato. No final, utilize um algoritmo de minimização de estados de automato a fim de minimizar o automato produzido pelo algoritmo de transformação.

Resposta: Esta linguagem é um conjunto de zeros e uns desde que se aparece zero ou um, aparece pelo menos dois. O DFA construído foi:



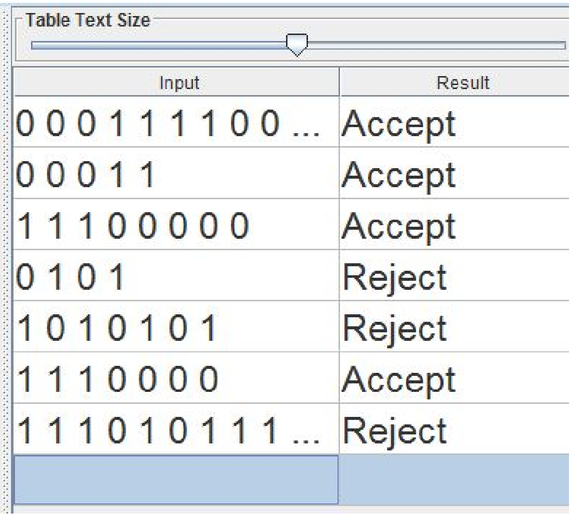


Table Text Size

Input	Result
0 0 0 1 1 1 1 0 0 ...	Accept
0 0 0 1 1	Accept
1 1 1 0 0 0 0 0	Accept
0 1 0 1	Reject
1 0 1 0 1 0 1	Reject
1 1 1 0 0 0 0	Accept
1 1 1 0 1 0 1 1 1 ...	Reject

Tabela de estados distinguíveis

q1	x			
q2	x	x		
q3	x	x	x	
q4	x	x	x	x
	q0	q1	q2	q3

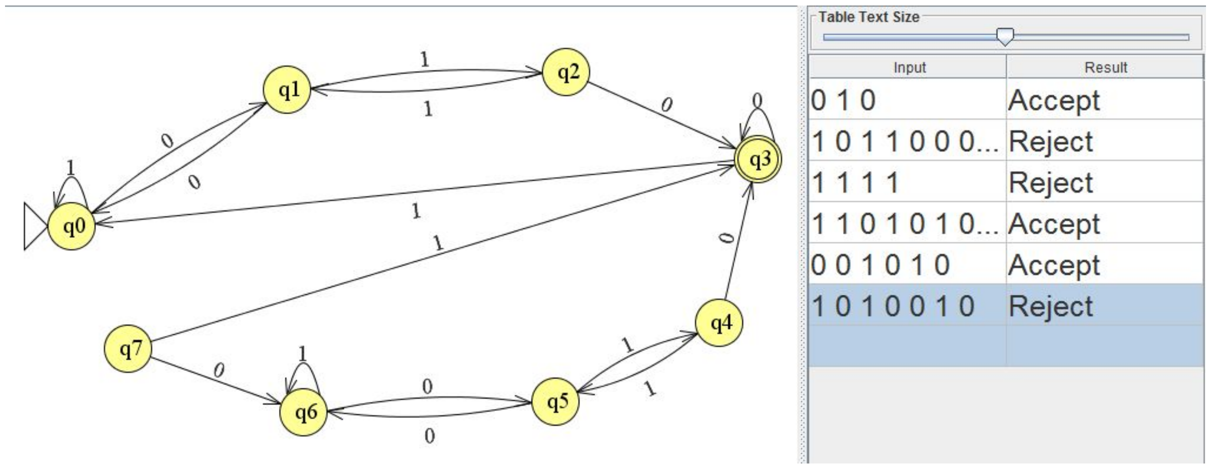
Todos os estados são distinguíveis, portanto o DFA já está minimizado.

Aula 07/04

1.) Construa o DFA com o número mínimo de estados

	0	**1**
->A	**B**	**A**
B	**A**	**C**
C	**D**	**B**
***D**	**D**	**A**
E	**D**	**F**
F	**G**	**E**
G	**F**	**G**
H	**G**	**D**

Esse é o DFA, contruído a partir das entradas da tabela



Agora, verificaremos se podemos minimizá- la construindo a tabela de equivalência de estados.

A								
B	X							
C	X	X						
D	X	X	X					
E	X	X	EQ	X				
F	X	EQ	X	X	X			
G	EQ	X	X	X	X	X		
H	X	X	X	X	X	X	X	
	A	B	C	D	E	F	G	H

Assim, temos os estados q0(A,G); q1(B,F); q2(C,E); q3(D); q4(H) minimizados, gerando o DFA mínimo abaixo, já que os estados são equivalentes, cada entrada irá gerar uma saída igual, ao original.

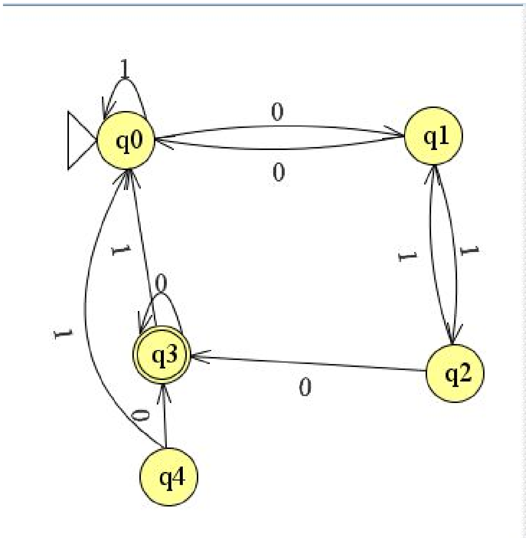


Table Text Size

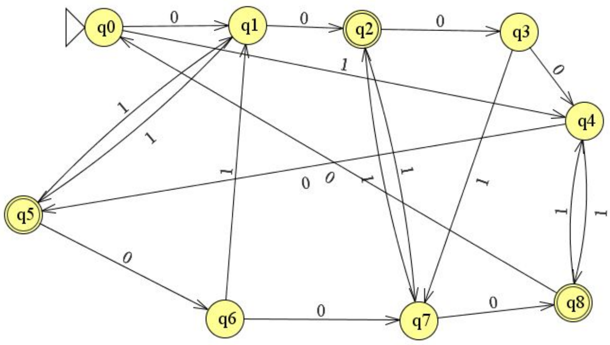
Input	Result
0 1 0	Accept
1 0 1 1 0 0 0 0 0 0	Reject
1 1 1 1	Reject
1 1 0 1 0 1 0 1 0	Accept
0 0 1 0 1 0	Accept
1 0 1 0 0 1 0	Reject

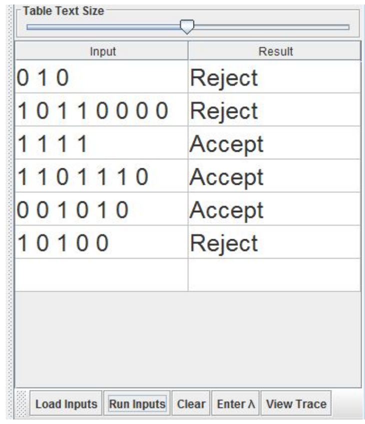
2.) Construa o DFA com o número mínimo de estados

	0	**1**
->A	**B**	**E**
B	**C**	**F**
***C**	**D**	**H**
D	**E**	**H**
E	**F**	**I**

***F**	**G**	**B**
G	**H**	**B**
H	**I**	**C**
***I**	**A**	**E**

Temos que o o DFA gerado com os dados da tabela é:

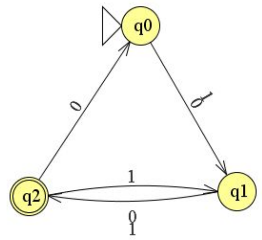




Verificaremos se é possível construir um DFA mínimo, construindo a tabela de equivalências:

A									
B	X								
C	X	X							
D	EQ	X	X						
E	X	X	X	X					
F	X	X	EQ	X	X				
G	EQ	X	X	EQ	X	X			
H	X	EQ	X	X	EQ	X	X		
I	X	X	EQ	X	X	EQ	X	X	
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**

Assim, na minimização, podemos criar os estados q0(A,D,G), q1(B,E,H), q2(C,F,I)

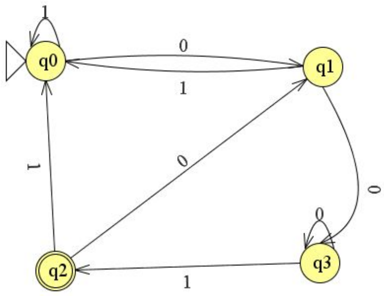


Input	Result
0 1 0	Reject
1 0 1 1 0 0 0 0	Reject
1 1 1 1	Accept
1 1 0 1 1 1 0	Accept
0 0 1 0 1 0	Accept
1 0 1 0 0	Reject

3.) Forneça um DFA mínimo que aceita todas as strings de 0's e 1's que termina em 001. Prove que é minimo.

Resolução: Para ser mínimo deve obedecer as seguintes propriedade:

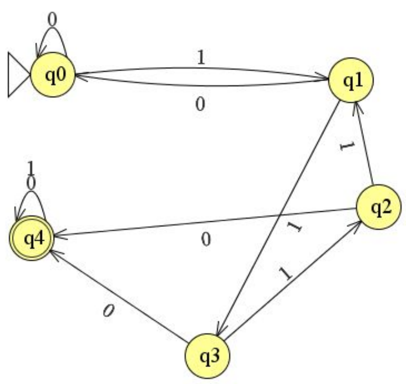
1. Deve ser Determinístico (DFA)
2. Não pode ter estados inacessíveis (não atingíveis a partir do estado inicial)
3. A função programa deve ser total (a partir de qualquer estado são previstas transições para todos os símbolos do alfabeto)



Input	Result
1 0 0	Reject
0 1 1 0 1 0 1 0 1 1 0 0 1	Accept
1 0 1 0 1 0 0 1	Accept
1 1 1 0 0 1	Accept
1	Reject
0 0 1	Accept
0 0 0 0 0 1 1 0	Reject

Como observamos pelas transições, o DFA obedece as propriedades citadas acima.

4.) Forneça um DFA mínimo que aceita todas as strings de 0's e 1's que contenha 110 como substring verifique que o automato seja mínimo



Input	Result
1 1 0 0 1 1 0 0	Accept
0 0 1 1 0 1 0 1	Accept
1 0 1 0 1 0 1 0	Reject
0 0 0 0 1 1 0 0	Accept
1 0 0 1 1 0 0 1 1	Accept
0 0 1 1	Reject
0 0	Reject
1 1 0	Accept

Tabela de Equivalências:

->q0					
q1	x				
q2	x	x			

q3	x	x	x		
q4	x	x	x	x	
	->q0	q1	q2	q3	q4*

Como não há nenhum estado que é equivalente ao outro, temos que o DFA mínimo será igual ao inicial

Aula 14/04

1. Prove que a linguagem L={w|w consiste em todos os strings com igual número de 0's e 1's} é não regular.
   Resposta:
   Usando Lema do Bombeamento: que segue as seguintes propriedades

1. u = xyz (x é um prefixo, z é um sufixo)
2. |y | ≥ 1 (a parte do meio y é não vazia)
3. |xy| ≤ p (bomb. nos p primeiros simbolos)
4. xy^iz ∈ L $\forall i \geq 1$

Seja $0^n\ 1^n$ uma das formas que aceitam as strings com número iguais de 0's e 1's. Consideremos $0^p\ 1^p$, por hipótese, a parte a ser bombeada deve estar contida nos p primeiros elementos da string.
Tomando como o caso base, tamanho de bombeamento p = 2:

0	0	1	1

x y z

xyz = 0011
temos que a string sera aceita para i = 1, mas se i >1:
0(00…)11, faz com que o número de 0's seja maior que o número de 1's, não satisfazendo a condição 4.

2. Tente provar que a linguagem (00+11)* é não regular.

Resposta: Usando o lema do bombeamento:
Temos por definição de fechamento, que a string a ser aceita pode ser {ε, 00, 11…}, para que seja uma linguagem regular, necessita que pelas condições citadas no exercício anterior, a string possa ser escrita da forma u = xyz,

- Se u = ε
- Implica que |y| <1, portanto. não satisfaz a condição 2

3. Seja ∑ = {0,1}. Se L1 ={0} e L2 = {1}*. Determine:

a) L1 ∪ L2 e L1 ∩ L2
Resposta: (0+1*) e ε
b) L1* e L2*
Resposta: 0* e (1*)*
c) (L1 ∪ L2)*
Resposta: (0+1*)*
d) (L1 ∩ L2)*
Resposta: ε
e) (∑*-L1)
Resposta: (00*1*) + (1*00*)
f) (∑*-L2) ∩ L1*
Resposta: 0*
g) $(L1.L2)^R$
Resposta: (L2.L1)

Aula 19/04

1. Gere o CFG:

a){$a^i b^j c^k$ / i ≠ j ou j ≠ k}
Resposta:
S -> AC | BC | DE | DF
A -> a | aA | aAb
B -> b| Bb| aBb
C->c | cC
D->a | aD
E-> b | bE |bEc
F-> c| Fc| bFc

b){w / w termina e começa com o mesmo símbolo}
Resposta:
(0(0+1)*0) + (1(0+1)*1)
S->A+A
A->0B| 1B | 0 | 1 | ε
B-> S*0| S*1

c){w / w é ímpar e seu símbolo do meio é 0}
Resposta:
R -> 0S1 | 1S1
S -> 0S0 | 0S1 | 1S0 | 1S1 | 0

d)Strings binárias com duas vezes mais 1's do que 0's
Resposta:
R -> R -> 011R | 01R1 | 0R11 | R011 | 110R | 11R0 | 1R10 | R110 | R101 | 101R | ε

2. A Gramática a seguir gera a linguagem regular: 0*1(0+1)*
S -> A1B
A -> 0A | ε
B -> 0B |1B | ε
Forneça a derivação mais a esquerda e mais a direita das strings

a.) 00101
Resposta:
Esquerda
S -> A1B
S -> 0A1B
S -> 00A1B
S -> 00ε1B
S -> 0010B
S -> 00101B
S -> 00101ε

Direita
S -> A1B
S -> A10B
S -> A101B
S -> A101ε
S-> 0A101
S -> 00A101
S -> 00ε101

b.) 1001
Esquerda:
S -> A1B
S -> ε1B
S -> 10B
S -> 100B
S -> 1001B
S -> 1001ε

Direita:
S -> A1B
S -> A10B
S -> A100B
S -> A1001B
S -> A1001ε
S-> ε1001

c.) 00011
Esquerda:
S -> A1B
S -> 0A1B
S -> 00A1B
S -> 000A1B
S -> 000ε1B
S -> 00011B
S -> 00011ε

Direita:
S -> A1B
S -> A11B
S -> A11ε
S -> 0A11
S -> 00A11
S -> 000A11
S -> 000ε11

Aula 26/04

1.) Construa CFG's que gerem as seguintes linguagens:

a.) {w / |w| é par e maior que 0}
Resposta:
S -> A11A | A10A | A00A | A01A
A -> 10A | A10 | A00 | 00A | A11 | 11A | A01 | 01A | ε

b.) {w / w contém um número par de 0's}
Resposta:
A -> 1A | R | E
R -> 00A | 0A0 | A00

c.) {w / w possui o 3º símbolo da direita para a esquerda igual a 1}
Resposta: (0+1)*1(0+1)(0+1)
1.S->E*1E
2.E->(E)| E+E| EE
3.E->1|0

d.) Strings que sejam comentários da forma /#(a+b)*#/
Resposta:
1. S-> /E/
2. E->#E#| (E) | A+A
3. A-> a | b

e.) {w / w possui um número igual de 0's e 1's}
Resposta:
1.S-> 0S1 | 1S0 | A01A | A10A | ε
2.A-> ε / 0A1A / 1A0A

Resposta:

2.) Mostre para toda linguagem regular há uma gramática regular
Resposta:
Para mostrar que uma RE possui uma CFG, basta mostrar que os operdores conctenação, união e fehamento são formados por uma CFG. Logo:

União: (a+b) = U ->a|b
Concatenação : ab = U->ab
Fechamento: a* = U->aU|ε

então qualquer RE pode ser transformada em uma CFG

3.) Monte uma CFG que reconhece double numa linguagem de programação
Resposta:
N->0| 1| 2|...|9
M->+| - | ε
E->MNF | ε
F->NF|ε
S->MNF.FE | MF.NFE | MNFE

Aula 28/04

1.) Construa um CFG's que gere as linguagens:

a) $\{a^i b^j c^k \ / \ i \neq j\}$

Resposta:

A → aAbb | aaAb | C | ε| aA | Ab

C → cC | ε

b) $\{a^i b^j c^k \ / \ i = j + k\}$

Resposta:

R → aRc | K | ε

K → aKb | ε

c) $\{a^{k+1} b \ c^{2k} \ / \ k >= 1\}$

Resposta:

B → bB | ε

C → aCcc | B

S → aC

d) $\{a^m b^m c^n d^n \ / \ m>=1, n >=1\}$

Resposta:

A → aAb | ε

B → cBd | ε

S → aAbcBd

e) $\{a^n b^m c^{2n+m} \ /m, n>0\}$

Resposta:

A → aRcc | B | ε

B → bBc | ε

C → aABccc

f) $\{a^m b^n \ / \ 0 <= n <= m <= 3n\}$

Resposta:

S→aSb | aaSb | aaaSb | ε

g) $\{a^m b^i a^n \ / \ i = m+n\}$

A -> aAb | ε

B -> bBa | ε

R -> AB

h) $\{a^n b^m c^i \ / \ 0 <= n +m <= 1\}$

Resposta:

A → aAc | B | ε

B → bB | ε

C → Ac

i) $\{a^n b^{2n} c^m$ / m, n >= 0}

Resposta:

A → aAbb | C

C → cC

Aula 03/05

1.) Seja G a gramática:

S -> abSc | A

A -> cAd | cd

a.) Dê uma derivação de ababccddcc

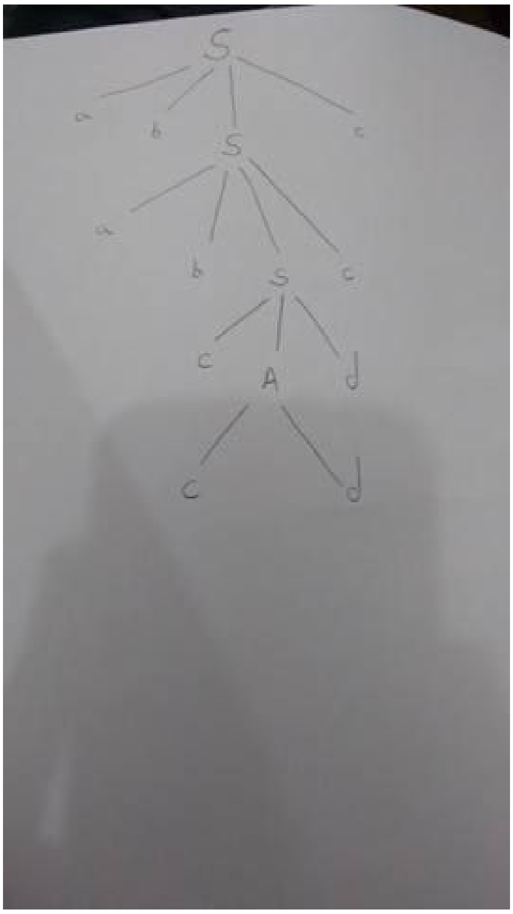
S -> abSc

S -> ababScc

S -> ababcAdcc

S -> ababccddcc

b.) Construa a árvore de análise sintática do item a



c.) Use a notação de conjunto para definir L(G).

Resposta: L(G) = $(ab)^i c^i d^i c^i$

2.) Seja G a gramática:

S -> ASB | ε

A -> aAb | ε

B -> bBa | ε

a.) Dê uma derivação mais à esquerda de aabbba

Resposta:

S -> ASB

S -> aAbSB

S -> aaAbbSB

S -> aaεbbSB

S -> aabbεB

S -> aabbbBa

S -> aabbbεa

b.) Dê uma derivação mais à direita de abaabbbabbaa

Resposta:

S -> ASB

S -> ASbBa

S -> ASbbBaa

S -> ASbbεaa

S -> AASBbbaa

S -> AASbBabbaa

S -> AASbεabbaa

S -> AAεbabbaa

S -> AaAbbabbaa

S -> AaaAbbbabbaa

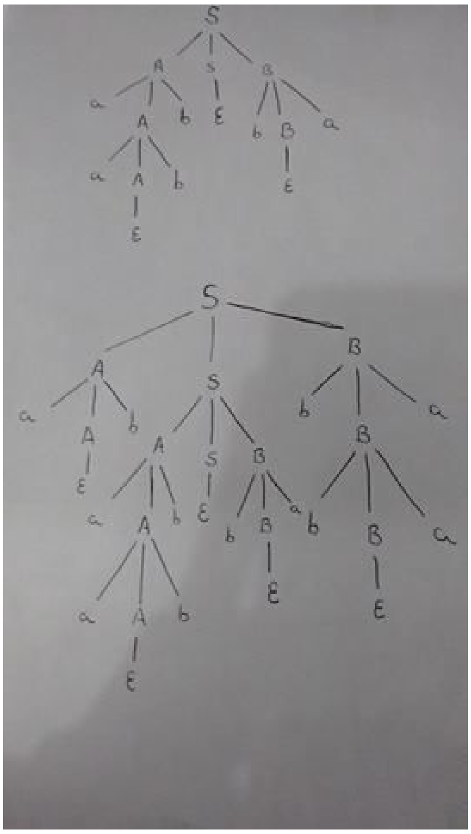
S -> Aaaεbbbabbaa

S -> aAbaabbbabbaa

S -> aεbaabbbabbaa

abaabbbabbaa

c.) Construa a árvore de análise sintática para as derivações de a e b



d.) Use a notação de consumo para definir L(G)
Resposta: L(G) = a(ab+ba)*a

3.) Seja G a gramática:

S -> SAB | ε
A -> aA | ε
B -> bB | ε

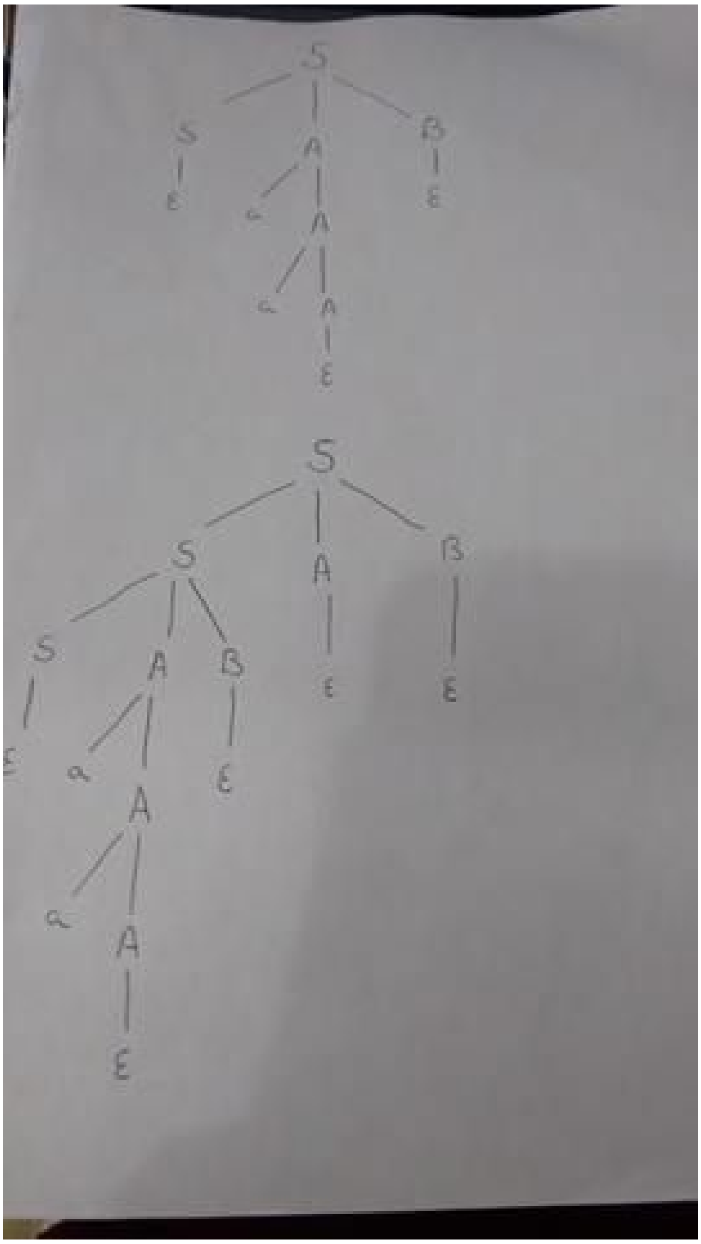
a.) Dê uma derivação à esquerda de abbaab
Resposta:
S -> SAB
S -> SABAB
S -> εABAB
S -> aABAB
S -> aεBAB
S -> abBAB
S -> abbBAB

S -> abbεAB
S -> abbaAB
S -> abbaaAB
S -> abbaaεB
S -> abbaabB
S -> abbaabε

b.) Dê duas derivações à esquerda de aa
Resposta:
S -> SAB
S -> εAB
S -> aAB
S -> aaAB
S -> aaεB
S -> aaε

S -> SAB
S -> SABAB
S -> εABAB
S -> aABAB
S -> aaABAB
S -> aaεBAB
S -> aaεBAB
S -> aaεAB
S -> aaεB
S -> aaε

c.) Construa as árvores de análise sintática de b



d.) Dê uma expressão regular de L(G)

Resposta: L(G) = (a* + b*)*

4.) Mostre que a gramática abaixo é ambígua:

S-> 0A | 1B

B-> 1BB | 0S | 0

A -> 0AA | 1S | 1

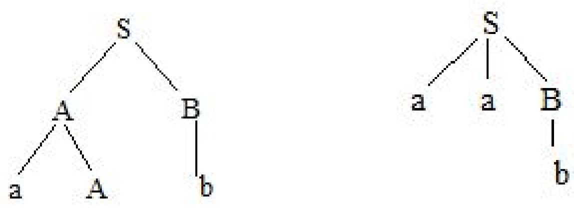
Resposta:
Basta mostrar duas derivações mais à esquerda que resultam em strings diferentes.
Derivação 1.
S -> 0A
S -> 0AA
S -> 01A
S -> 011
Derivação 2.
S -> 0A
S -> 0AA
S -> 01SA
S -> 010AA
S -> 0101
S -> 01011

5.) Dado a gramática
S->AB | aaB
A-> a | aA
B-> b

a.) Dê duas derivações mais à esquerda para aab.
Resposta:

S->AB	S->aaB
S->aAB	S->aab
S->aab	

b.) Dê as duas árvores de análise sintática



c.) Encontre uma gramática ambígua.
Resposta: S -> (S)S | ε

Aula 12/05

1.) Construa um PDA para as linguagens abaixo:

a.) Strings com número igual de 0’s e 1’s

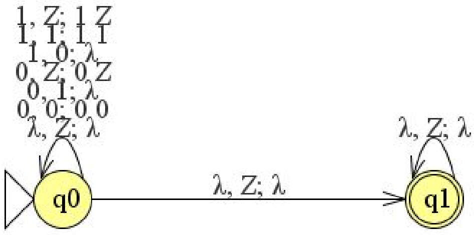


Table Text Size

Input	Result
0 1 1	Reject
1 0 1 0 1 0 0 1	Accept
0 1 0 1 0 1 0 0	Reject
0 1 1	Reject
1 0 0 0 0 0 1	Reject
1 1 0 0	Accept
0 1	Accept
1 0 0 1	Accept

Load Inputs | Run Inputs | Clear | Enter λ | View Trace

b.) {$0^n \ 1^{2n}$ | n $\geq$0}

Automaton Editor | Multiple Run

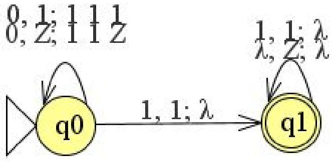


Table Text Size

Input	Result
0 1 1	Accept
0 0 0 1 1 1 1 1 1	Accept
0 1 0 1 0 1 0 0	Reject
0 1 1	Accept
1 0 0 0 0 0 1	Reject
1 1 0 0	Reject

c.) $\{ww^r$ | w está em (0+1)*}

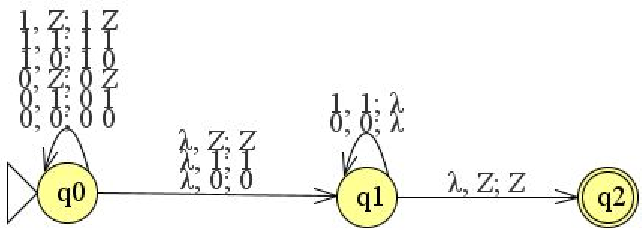


Table Text Size

Input	Result
1 1 1 1	Accept
1 0 1 0 1 0	Reject
1 0 0 0 0 1	Accept
1 0 1 1 0 1	Accept
0 0 1 1 0 0	Accept
0 1 0 1 0 1 1 ...	Reject

Load Inputs | Run Inputs | Clear | Enter λ | View Trace

d.) $\{a^n b^n c^m d^m \mid n \geq 0$ e $m \geq 1\}$

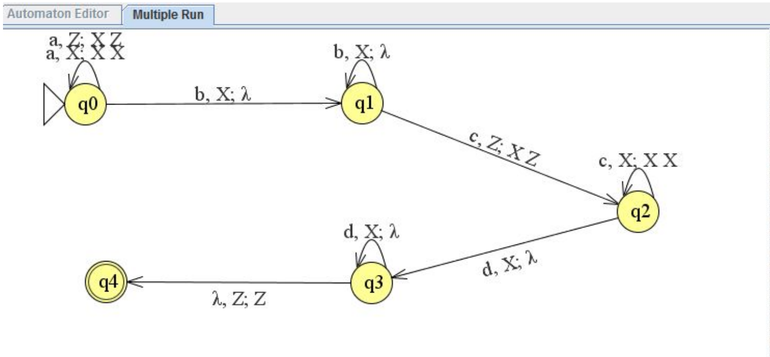


Table Text Size

Input	Result
a a b b c c c d d d	Accept
a b c d	Accept
a a b c c d d	Reject
a b b b b c c c a d	Reject
a b c c c d d d	Accept
a b c d d	Reject
a a b b c d	Accept

Aula 17/05

1.) Construa PDA's p/ as linguagens:

a.) $a^i\, b^j\, c^k \setminus k = 2(i+j)$

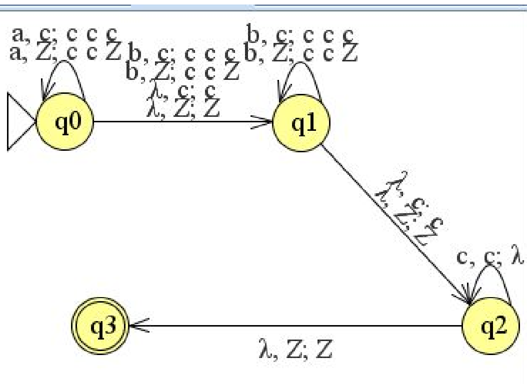
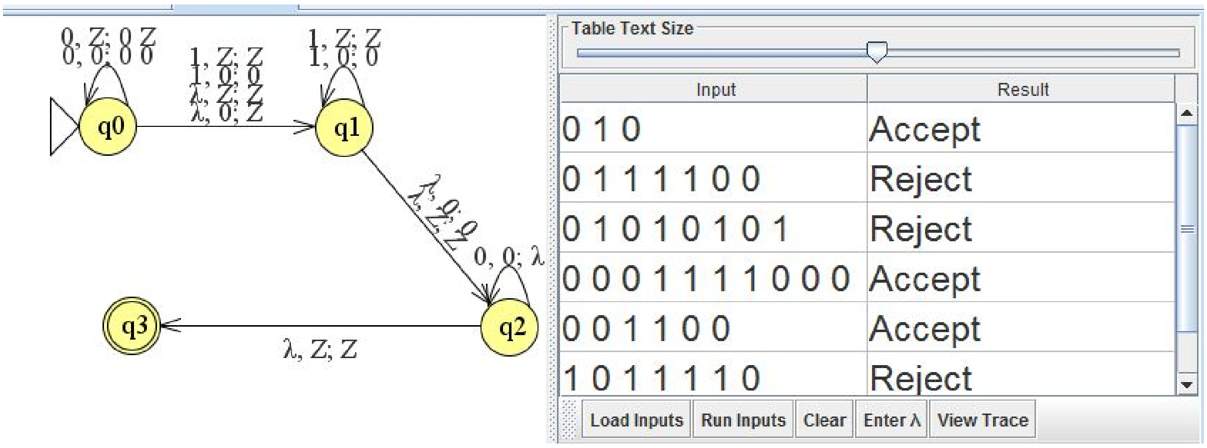


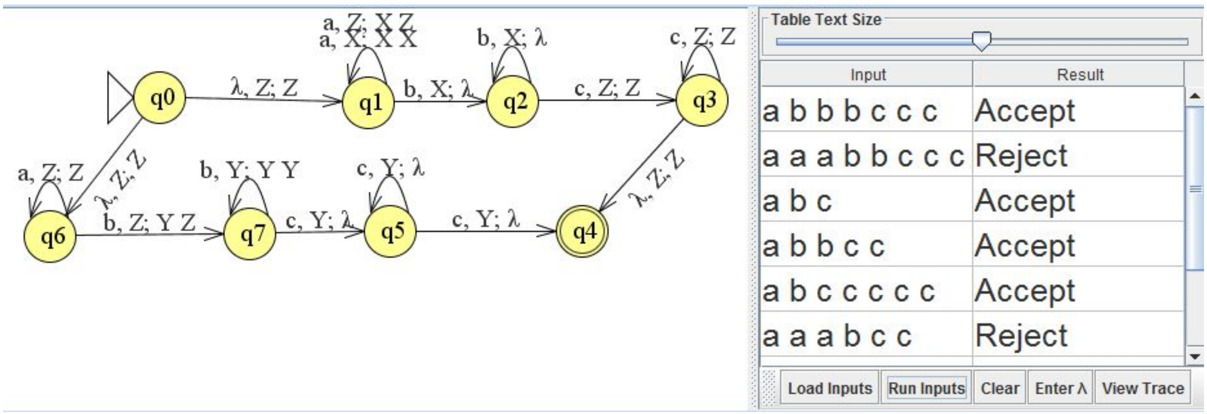
Table Text Size

Input	Result
a b c c c c	Accept
b c c	Accept
a b c	Reject
a b b c	Reject
a b b c c c c c c	Accept
a a b c c c	Reject

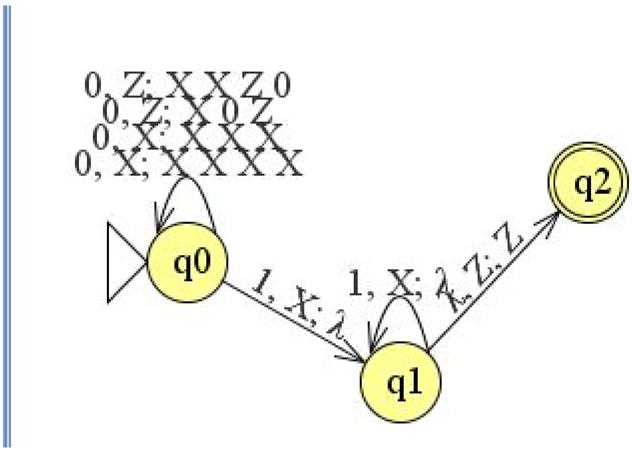
Load Inputs | Run Inputs | Clear | Enter λ | View Trace b.) $0^m1^n0^m \setminus m,\ n >= 0$



c.) $a^ib^jc^k \setminus i = j\ ou\ j = k$

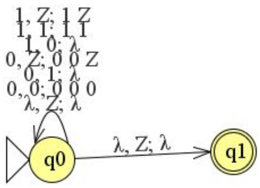


d.) $a^n1^k \setminus n <= k < 2n$ a1



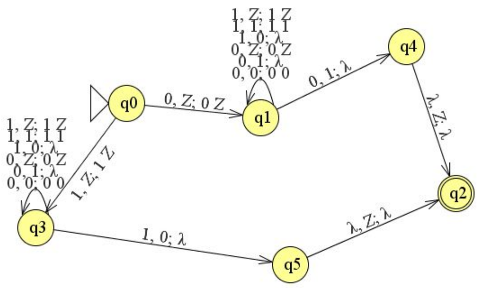
2.) Construa PDA’s determinísticos para:

a.) Strings binários com duas vezes mais 1’s que 0’s



0 1 1	Accept
1 1 0 0 0 0	Reject
0 1 0 1 0 1 1 1 1	Accept

b.) Strings binárias que começam e terminam com o mesmo símbolo e tem o mesmo número de 0’s e 1’s.



Input	Result
0 1	Reject
0 0 0	Reject
1 1 1 1 1	Reject
1 0 0 1	Accept

3)Apresente um exemplo de ID para cada PDA dos exercícios 1 e 2.

1.

a)String **abcccc**

$(q_o, abcccc, z_o)$

$(q_1, bcccc, az_o)$

$(q_2, cccc, abz_o)$

(q_2, ccc, abz_o)

(q_2, cc, bz_o)

(q_2, c, bz_o)

$(q_2, , z_o)$

$(q_3, , z_o)$

b)String **010**

$(q_0, 010, z_o)$
$(q_1, 10, 0z_o)$
$(q_2, 0, 0z_o)$
$(q_2, , z_o)$
$(q_3, , z_o)$

c)String **bc**

(q_0, bc, z_0)
(q_1, bc, z_0)
$(q2, c, bz_0)$
$(q_3, , z_0)$
$(q_4, , z_0)$

d)String **a1**

$(q_0, a1, z_0)$
$(q_0, 1, az_0)$
$(q_1, , z_0)$
$(q_2, , z_0)$

2.a) String **011**

(q0, 011, z0)
(q0, 11, 00z0)
(q0, 1, 0z0)
(q0, , z0)
(q1, , z1)

b.) String **1001**

(q0, 1001, z0)
(q3, 001, 1z0)
(q3, 01, z0)
(q3, 1, 0z0

Aula 19/05

1.) Responda:

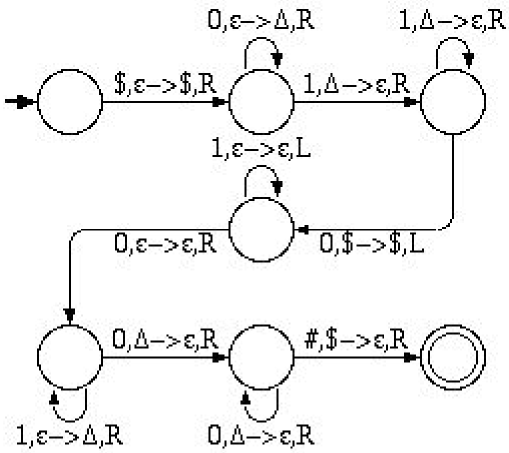
a.) Em que casos se diz que uma gramática é ambígua?
Resposta: Quando a gramática gera pelo menos uma sentença que possui 2 ou mais sequências de derivações, sendo elas mais à esquerda ou mais a direita, ou também quando possui mais de uma árvore de derivação distinta.

b.) Em que casos se diz que uma linguagem é ambígua? Dê um exemplo.
Resposta: Ocorre quando não é possível evitar a geração de strings ambíguas de uma gramática sem alterar a linguagem.
Um exemplo comum de ambiguidade em linguagens de programação são os condicionais se-então-senão.

S → se b então S
 senão S
S → se b
 então S
S → p

c.) Há interesse prático em se trabalhar com gramática ambígua?
Resposta: Não pois um computador é determinístico e uma gramática ambígua poderia gerar erros de sistema.

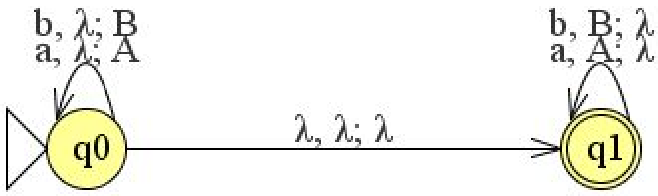
2.) Mostre que a linguagem $0^n1^n0^n \setminus n > 0$ pode ser reconhecida por um PDA que se move tanto para a direita quanto para a esquerda (ou um PDA com duas pilhas).



Esse PDA de duas pilhas trabalha movendo para a direita através do string para certificar-se que ele começa com 0^n 1^n . Então ele se move para a esquerda, até o início dos 1s e continua para a direita, para verificar 1^n 0^n .

Aula 24/05

Dado o PDA abaixo:



a.) Faça a tabela de transição de M

δ(q0,b,)={q0,B}

δ(q0,a,)={q0,A}

δ(q0,ε,)={q1,ε}

δ(q1,a,A)={q1,ε}

δ(q1,b,B)={q1,ε}

b.) Apesente todas as computações de abb
Resposta:

(q0,abb,z0)
(q0,bb,Az0)
(q1,b, BAz0)
(q1, , Bz0)

c.) Mostre que baab e L(M)
Resposta:

(q0,baab,z0)
(q0,aab,Bz0)
(q1,ab, ABz0)
(q1, b, Bz0)
(q1, , z0)

Chegamos a um estado final e toda a string foi consumida. Logo baab pertence a L(M).

d.) Mostre que aaa não pertence L(M)
Resposta:
(q0, aaa, z0)
(q1,aa, Az0)
(q1, a, z0)

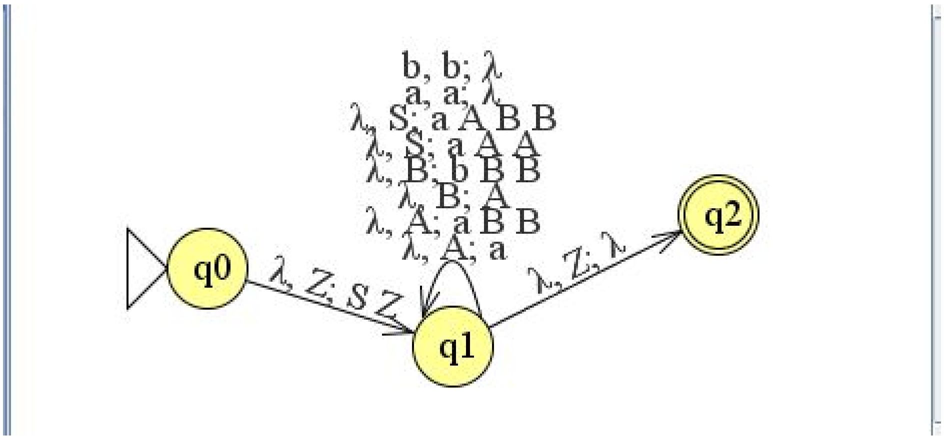
Como chegamos a um estado que não possui transição, ou seja um estado inválido a string aaa não pertence a L(M).

2.) Construir um PDA correspondente à gramática

S-> aABB / aAA

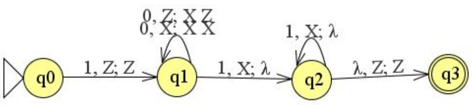
A-> aBB / a

B -> bBB / A

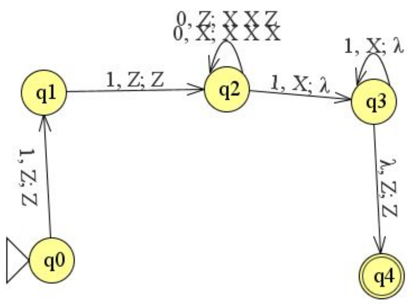


3.) Mostre que L = $\{1\ 0^n\ 1^n \setminus n > 0\} \cup \{110^n 1^{2n}\}$ é determinística

Resposta: Basta construir um PDA determinístico para cada parte da união.



Input	Result
0 1 1	Reject
1 1 0 0 0 0	Reject
0 1 0 1 0 1 1 1 1	Reject
1 0 0 1 1	Accept



Input	Result
0 1 1	Reject
1 1 0 0 0 0	Reject
0 1 0 1 0 1 1 1 1	Reject
1 0 0 1 1	Reject
1 1 0 1 1	Accept

4.) Prove que L = $a^i b^{2i} c^j \setminus i,j >= 0$ é livre de contexto

Resposta: Basta construir um CFG que represente a linguagem. CFG abaixo:

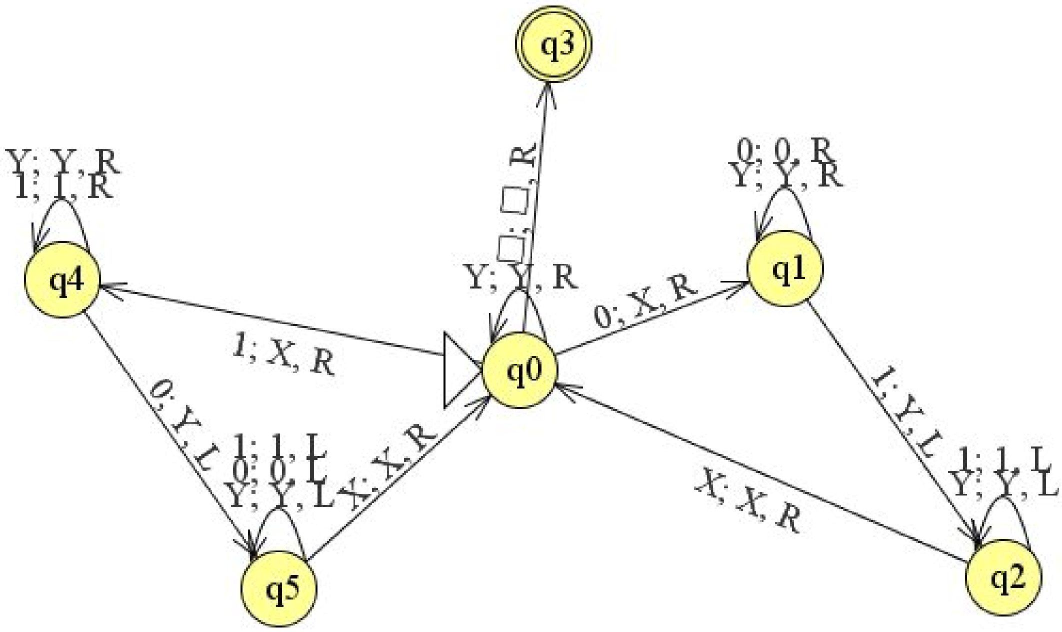
A -> aAbb | E
C -> cC | E
S -> AC

5.) Prove que a linguagem L = $1^k 0^i 0^j 1^j 0^k \setminus i,j,k > 0$ é livre de contexto
Resposta: Baste construir um CFG que represente a linguagem? CFG abaixo:
K -> 1K0 | 0H0W1
W -> 0W1 | E
H -> 0H | E

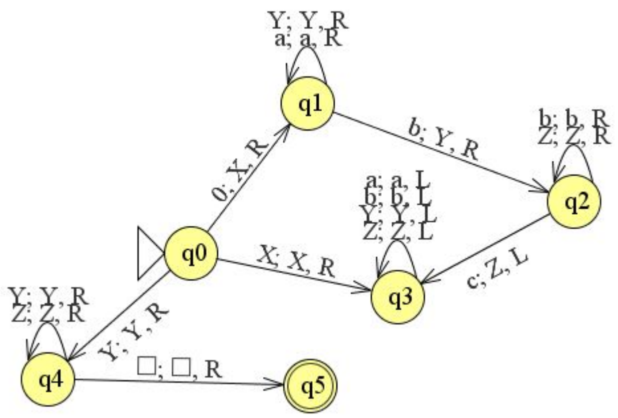
Aula 31/05

1. Projete máquinas de turing para as seguintes linguagens:

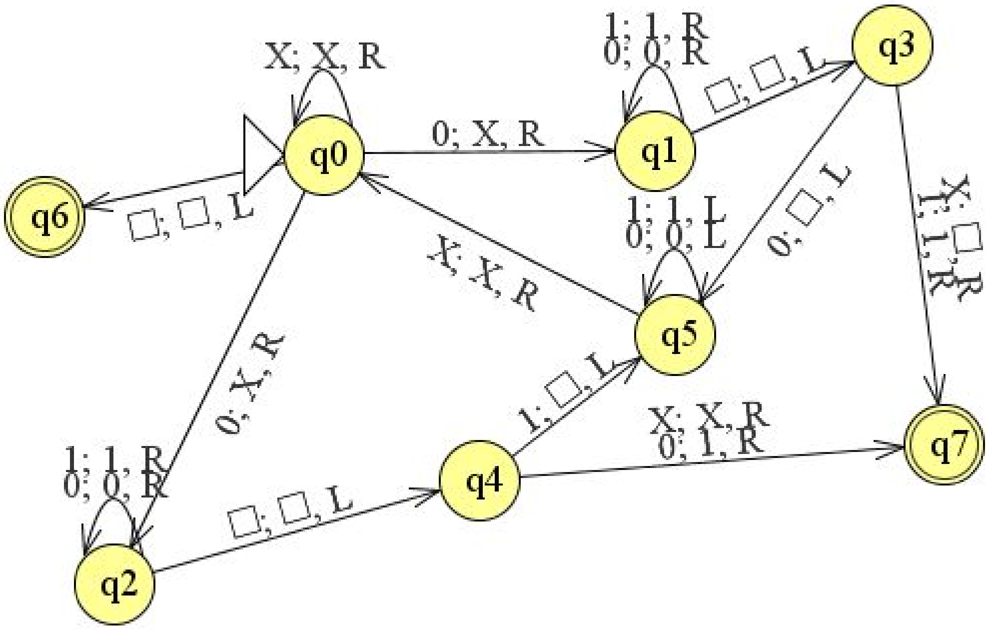
a.) Conjunto de strings com um número igual de 0’s e 1’s.



b.) $\{a^n b^n c^n \setminus n >= 1\}$



c.) {$ww^r \setminus w$ é qualquer string de 0’s e 1’s}



2.Considere a máquina de Turing M = ({q0,q1,q2,qf}, {0,1}, {0,1,B},, q0, B, {qf})

Descreva de maneira informal, mas clara, a linguagem L(M) se consistir nos seguintes conjuntos de regras:

a) δ(q0, 0) = (q1, 1, R); δ(q1, 1)= (q0, 0,R); δ(q1, B)= (qf, B,R);

Resposta: Troca os 0’s e 1’s de uma string. (troca 0 por 1, e 1 por 0, apenas uma vez).

b) δ(q0, 0) = (q0, B,R); δ(q0,1)= (q1,B,R); δ(q1, 1)= (q1,B,R); δ(q1, B) = (qf, B,R);

Resposta: Apaga uma string, trocando todos seus caracteres para branco.

c) δ(q0, 0) = (q1, 1, R); δ(q1, 1) = (q2, 0, L); δ(q2, 1) = (q0,1, R); δ(q1, B) = (qf, B,R);

Resposta: Percorre a string trocando sequências de 01`s por 10`s.

Aula 02/06

1. Mostre as IDs da máquina de Turing da linguagem $0^n 1^n$, tal que n1, se a fita contém de entrada:

A máquina de turing pode ser representada como:

$\delta(q_0,0)=(q_1, x, R)$
$\delta(q_1,0)=(q_1, 0, R)$
$\delta(q_1,y)=(q_1, y, R)$
$\delta(q_1,1)=(q_1, y, L)$
$\delta(q_2,y)=(q_2, y, L)$
$\delta(q_2,0)=(q_2, 0, L)$
$\delta(q_2,x)=(q_0, x, R)$
$\delta(q_0,y)=(q_3, y, R)$
$\delta(q_3,B)=(q_4, B, R)$

(a) 00

Resposta: $q_0$00 |- x$q_1$0 |- x0q_1 |- q_2x0|- x$q_0$0 |- xxq_1
Como não existe $\delta(q_1,B)$ 00 não pertence à linguagem

(b) 000111
Resposta:
$q_0$000111 |- x$q_1$00111 |- x0q_1 0111|- x00q_1 111|- x0q_2 0y11|- xq_2 00y11|- q_2 x00y11|-
xq_0 00y11 |- xxq_1 0y11 |- xx0q_1 y11|- xx0yq_1 11|- xxq_2 0yy1 |- xq_2 x0yy1|- xxq_0 0yy1 |-
xxxq_1 yy1 |- xxxyq_1 y1 |- xxxyyq_1 1 |- xxxyq_2 yy|- xxxq_2 yyy |- xxq_2 xyyy |- xxxq_0 yyy |-
xxxyq_3 yy |- xxxyyq_3 y |- xxxyyyq_3 |- xxxyyyq_4
Portanto a string é aceita pela linguagem

(c) 00111

Resposta:

q_0 00111|- xq_1 0111 |- x0q_1 111 |- xq_2 0y11 |- q_2 x0y11 |- xq_0 0y11 |- xxq_1 y11 |- xxy q_1 11 |- xxq_2 yy1 |- xq_2 xyy1 |- xxq_0 yy1 |- xxy$q_3 y1$ |- xxyyq_3 1

Como não foi atingido um estado de aceitação, a string não pertence à linguagem

1. Considere M a máquina de Turing definida por:

	B	a	b	c
q0	(q1,B,R)			
q_1	(q_2,B,L)	(q_1,a,R)	(q_1,c,R)	(q_1,c,R)
q_2	-	(q_2,c,L)	-	(q_2,b,L)

(a) Faça a computação da string aabca

Resposta:

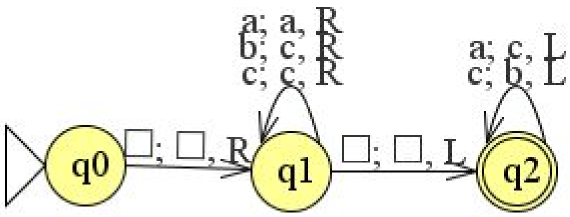
q0BaabcaB |- Bq1aabcaB |- Baq1abcaB |- Baaq1bcaB |- Baacq1caB |- Baaccaq1B |- Baaccaq1B |- Baacq2aB |- Baacq2ccB |- Baaq2cbcB |- Baq2abbcB |- Bq2acbbcB |- q2BccbbccB

(b) Faça a computação da string bcbc

Resposta:

q0BbcbcB |- Bq1bcbB |- Bcq1cbcB |- Bccq1bcB |- Bccccq1B |- Bccccq1B |- Bcccq2cB |- Bccq2cbB |- Bcq2cbbB |- Bq2cbbbB |- q2BbbbbB

(c) Dê o diagrama e transição de M

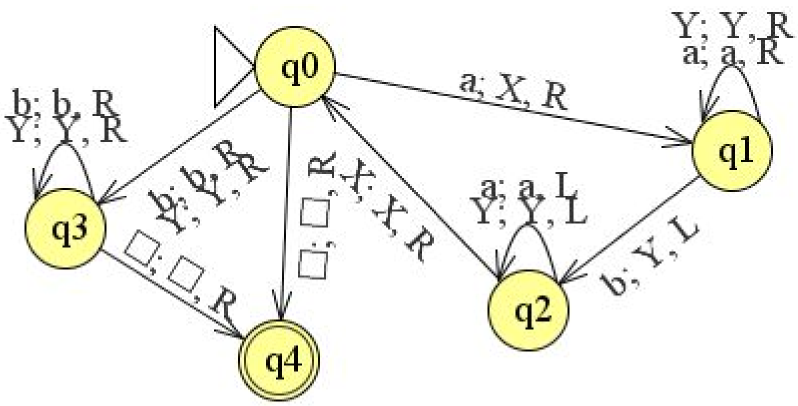


(d) Descreva o resultado de uma computação de M

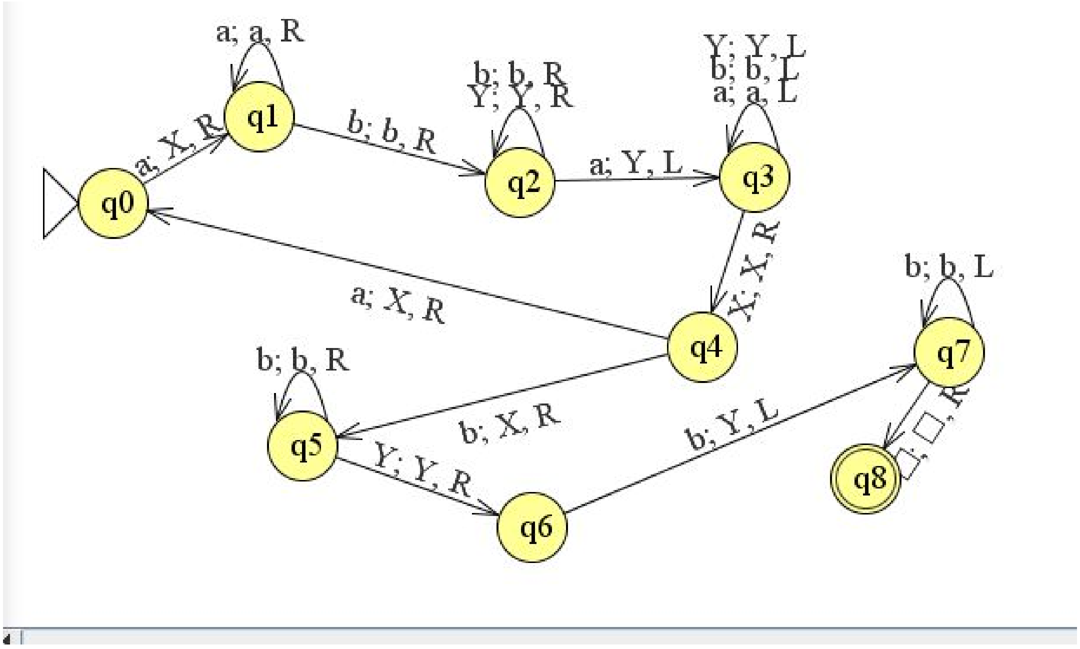
Resposta: Troca toda letra *a* pela letra *c* e toda letra *c* pela letra *b.*

3. Construa uma máquina de Turing com alfabeto {a,b} para aceitar cada linguagem a seguir por estado final:

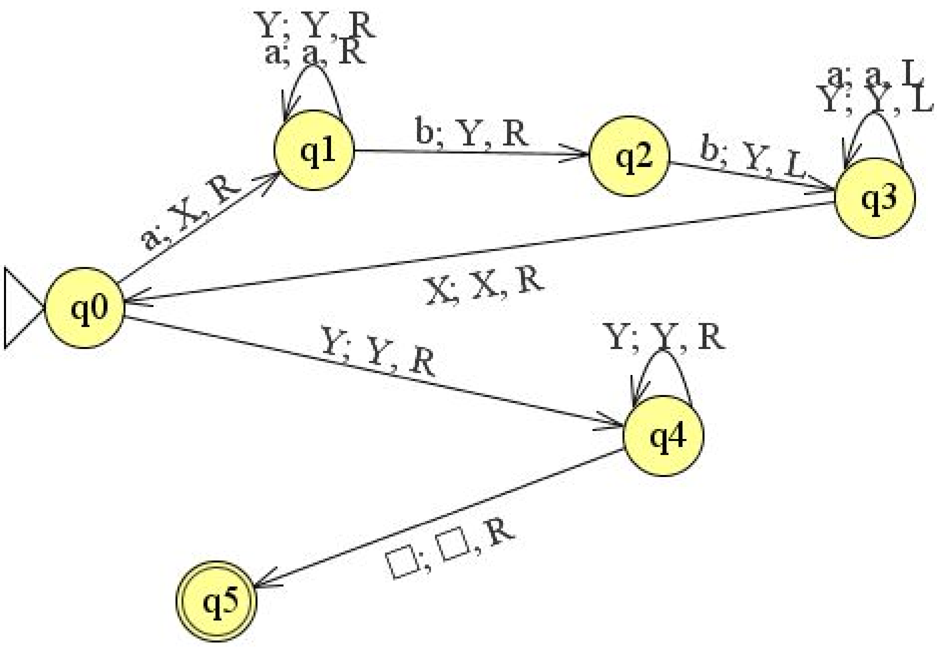
(a) $a^i b^j \setminus i >= 0, j >= i$



(b) $a^i b^j a^i b^j \setminus i, j > 0$



(c) $a^n b^{2n} \setminus n > 0$



Aula - Semana do Congresso

1. Construa uma TM que tenha como entrada um número N e adicione 1 para este número em binário. Para ser preciso, a fita inicialmente contêm um $ seguido por N em binário. A cabeça está inicialmente lendo o $ no estado q0. Sua TM deverá parar com N+1, em binário, gravado na fita, lendo o símbolo mais a esquerda de N+1, no estado qf. Você pode destruir o $ para criar N+1, se necessário.

Exemplos: q0$10011 ⊢* $qf10100 e q0$11111 ⊢* $qf100000

(a) Dê o diagrama de transição da sua TM e explique o propósito de cada estado.

Resposta:

Estado	0	1	$	B
->q0	(q0,0,R)	(q0,1,R)	(q0,$,R)	(q1,B,L)
q1	(q2,1,R)	(q1,0,L)	(q2,1,R)	-
*q2	-	-	-	-

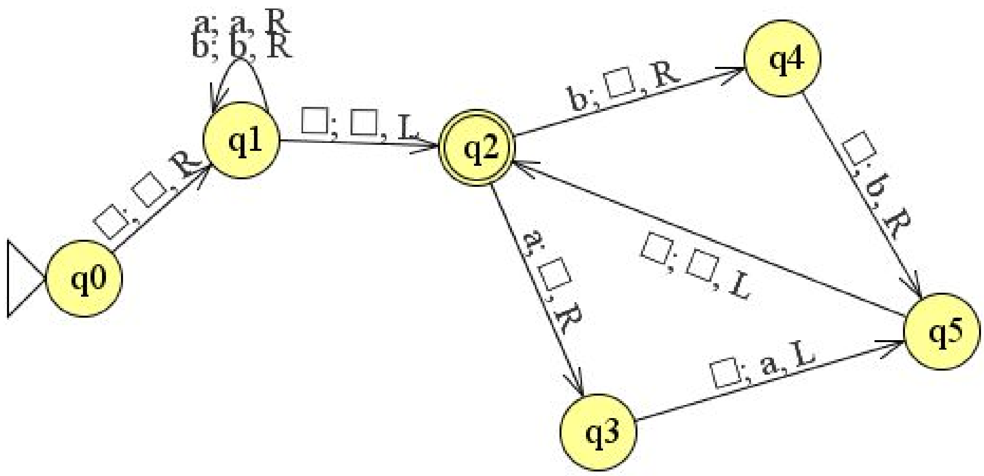
(b) Mostre a sequência de IDs da sua TM para a seguinte entrada: $111

Resposta:

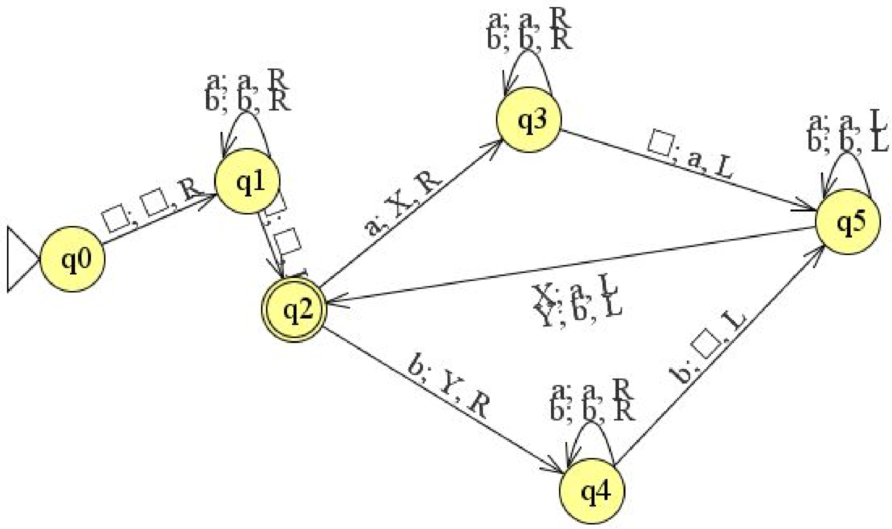
Q0$11 |- $Q011 |- $1Q011 |- $11Q01 |- $111Q0 -| $11Q11 -| $1Q110 |- $Q1100 |- Q1$000 |- 1Q2000

2. Construa uma TM sobre o alfabeto {a,b} para realizar as seguintes operações:

(a) Mover a entrada uma célula para a direita (Exemplo: q0BwB ⊢* qfBBwB).



(b) Concatenar uma cópia do reverso de uma string de entrada (Exemplo: q0BwB ⊢* qfBwwrB).



(c) Apagar os "b"s de uma entrada (Exemplo: q0BbabaababB ⊢* $qfBaaaaB).