UNIVERSIDADE FEDERAL DE VIÇOSA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

PEDRO CARDOSO DE CARVALHO MUNDIM - 3877

Trabalho Prático 1: Construindo Autômatos - Fundamentos da Teoria da Computação (CCF 131)

Primeiro Trabalho Prático da disciplina Fundamentos da Teoria da Computação - CCF 131, do curso de Ciência da Computação da Universidade Federal de Viçosa - Campus Florestal

Professor: Daniel Mendes Barbosa

FLORESTAL
2021

SUMÁRIO

1. Introdução

Este trabalho tem como foco a construção de autômatos finitos determinísticos ou não determinísticos, a fim de aplicar os conhecimentos adquiridos na disciplina até o presente momento. Para realizar tais construções, será utilizada a ferramenta JFLAP.

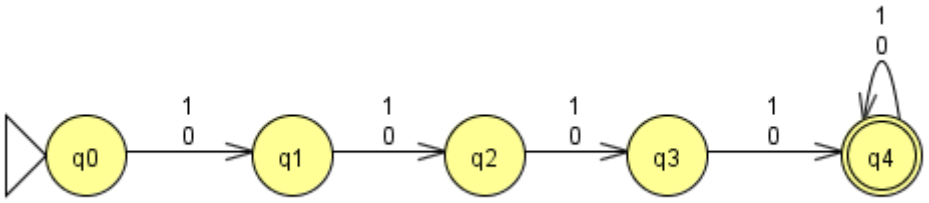
2. Desenvolvimento - Questões

Aqui, serão implementados os diagramas das máquinas de estados, de acordo com cada exercício solicitado. Além disso, serão mostrados diversos inputs e outputs para cada um deles, dentre os quais, 3 inputs apresentados serão aceitos pelas máquinas e 3 serão rejeitados.

Questão 01

Linguagem: O conjunto das palavras de tamanho maior do que 3.

Diagrama de Estados:



Resultado das Computações:

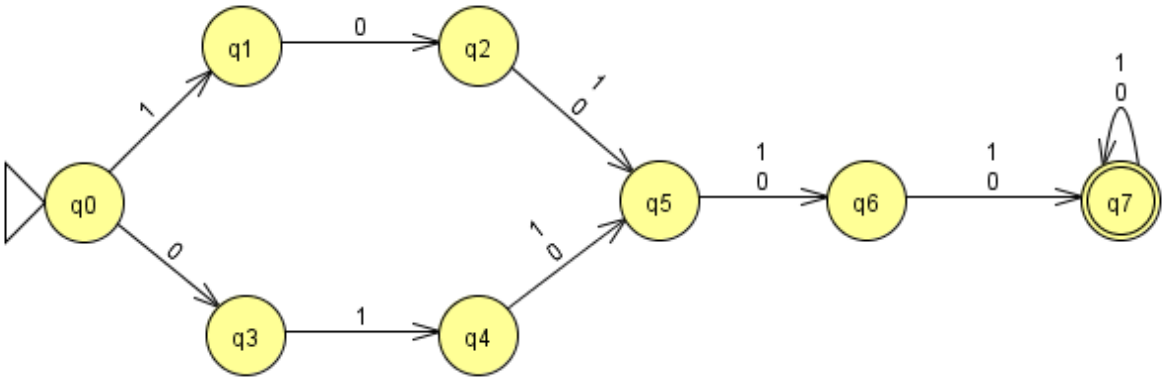
Input	Result
0000	Accept
1010	Accept
1110	Accept
101	Reject
11	Reject
00	Reject

Como precisamos de palavras com tamanho maior que 3, então é necessário pelo menos 4 estados. Como não temos restrições, o alfabeto pode ser qualquer um (0's e 1's nesse caso).

Questão 02

Linguagem: {w ∈ (0,1)* | |w| >= 5, e cujos dois primeiros símbolos sejam sempre diferentes um do outro.

Diagrama de Estados:



Resultado das Computações:

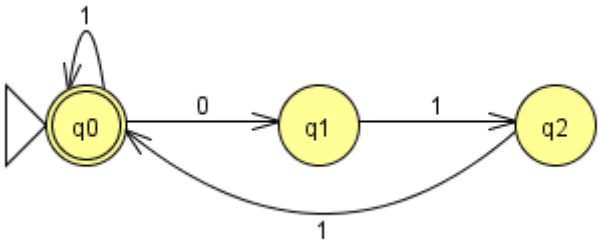
Input	Result
010000	Accept
101111	Accept
101010	Accept
110000	Reject
1010	Reject
0101	Reject

A primeira restrição é que os dois primeiros símbolos sejam diferentes. Para isso, criamos 4 estados após o estado inicial: se a palavra começa com “1”, cria-se a transição seguinte com “0”, obrigando então que tenhamos “10”. Se começar com “0” faremos o oposto, tendo então “01” (tendo aqui dois possíveis “caminhos”). Por fim, garantimos que a palavra tenha tamanho maior ou igual a 5. Para isso criamos então no mínimo 6 estados.

Questão 03

Linguagem: {w ∈ (0,1)* | cada 0 de w é imediatamente seguido de, no mínimo, dois 1's}.

Diagrama de Estados:



Resultado das Computações:

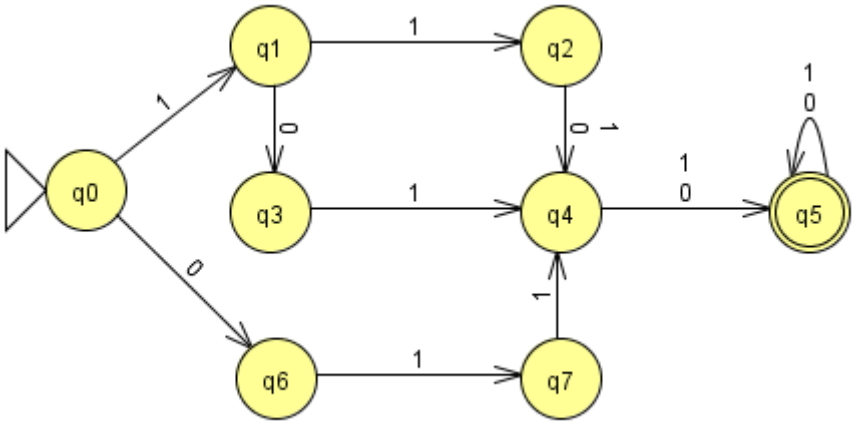
Input	Result
011	Accept
1011	Accept
10111011	Accept
101	Reject
1101	Reject
10110	Reject

Cada 0 deve ser acompanhado por dois "1's" (no mínimo). Então, se um "0" for lido, obrigatoriamente deve-se seguir para os estados q1, q2 e q0. Assim a condição é satisfeita.

Questão 04

Linguagem: **{w ∈ (0,1)* | |w| > 3 e os primeiros 3 símbolos de w contêm, no mínimo, dois 1's}**

Diagrama de Estados:



Resultado das Computações:

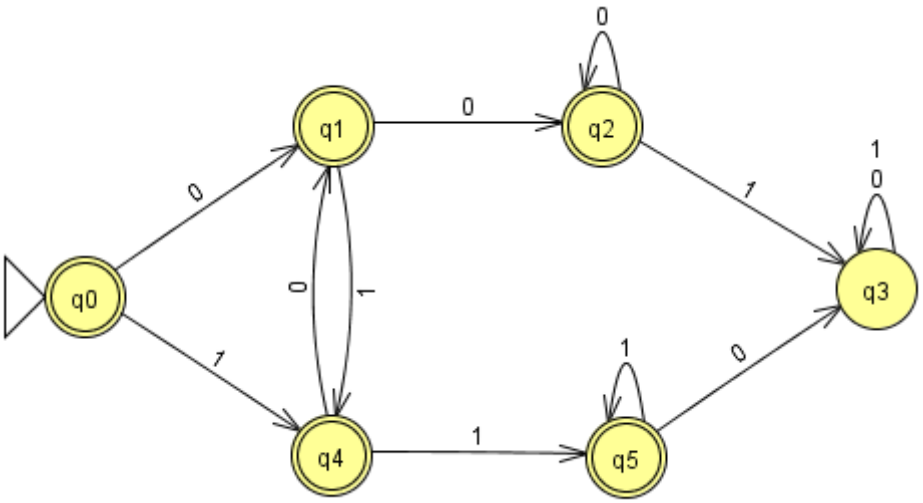
Input	Result
0111	Accept
0110	Accept
1101	Accept
1000	Reject
0001	Reject
0101	Reject

Primeiramente, garantimos que a palavra tenha tamanho maior que 3. Para isso, foi preciso pensar como garantir que dois "1's" aparecessem dentre os três primeiros símbolos. Por conta disso, foi preciso utilizar 4 transições e pelo menos 5 estados, sendo que no estado final é permitido mais leituras, gerando assim um laço sobre ele.

Questão 05

Linguagem: {w ∈ (0,1)* | w não possui a subpalavra 110 e nem a subpalavra 001}

Diagrama de Estados:



Resultado das Computações:

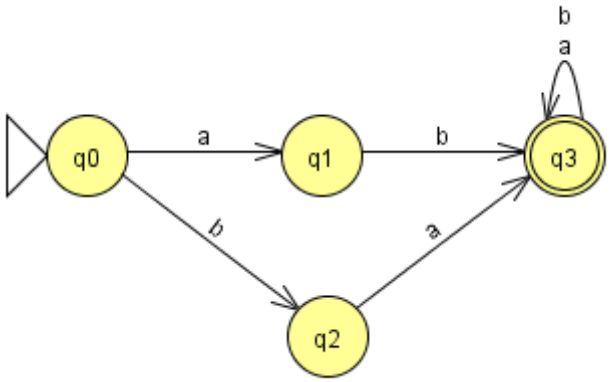
Input	Result
0111	Accept
101	Accept
100000	Accept
110	Reject
001	Reject
0110	Reject

A estratégia aqui foi utilizar um diagrama de estados que orienta-se pelo estado de erro. Quando houver uma subpalavra “00”, ao ler “1” ela irá para esse estado de erro. O mesmo acontece se houver a leitura do “11” e em seguida a leitura de um “0”.

Questão 06

Linguagem: {w ∈ (a,b)* | w sempre começa com dois símbolos diferentes e |w| > 1}

Diagrama de Estados:



Resultado das Computações:

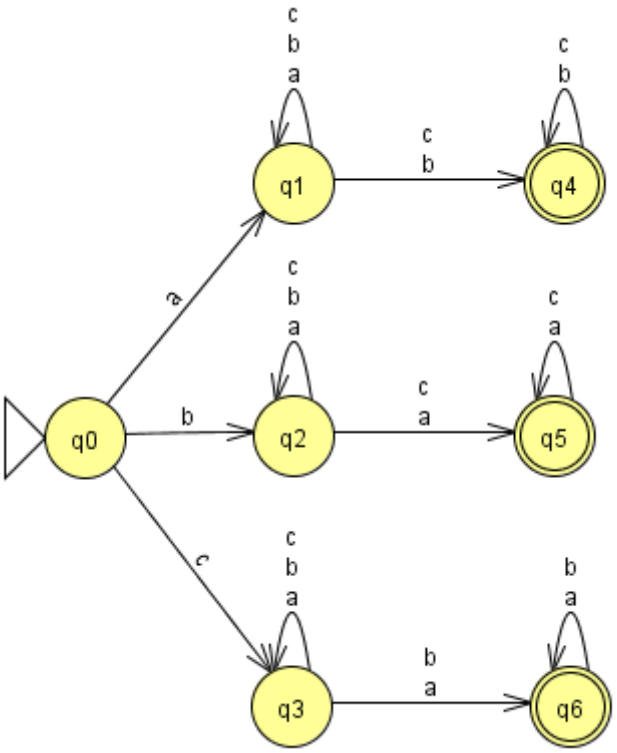
Input	Result
abb	Accept
bababa	Accept
abab	Accept
aa	Reject
bba	Reject
bbbbbbb	Reject

Queremos palavras com tamanho maior que 1 e o alfabeto é (a, b). Se o primeiro símbolo lido for “a”, então obrigatoriamente o próximo precisa ser um “b” e em seguida podemos ter mais quantos símbolos quisermos. Caso o primeiro símbolo lido for “b”, então obrigatoriamente o próximo precisa ser um “a”. Novamente, podemos ter em seguida quantos símbolos quisermos.

Questão 07

Linguagem: **{w ∈ (a,b,c)* | o primeiro símbolo de w é sempre diferente do último}**

Diagrama de Estados:



Resultado das Computações:

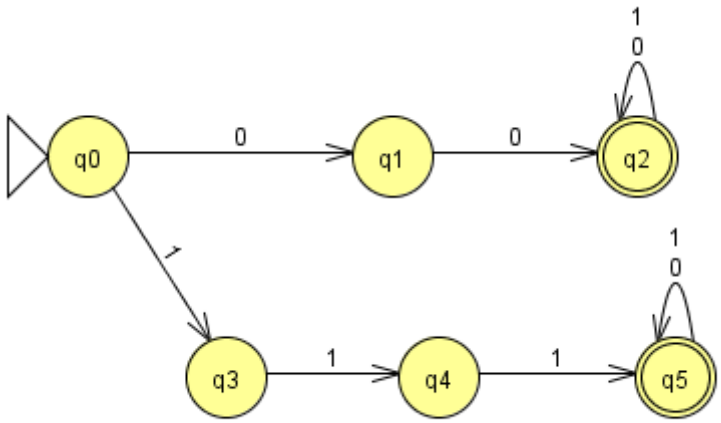
Input	Result
abac	Accept
aaaab	Accept
cbca	Accept
aa	Reject
cbbbbc	Reject
baaaab	Reject

Como agora o alfabeto é composto por (a, b, c) temos três possibilidades para o primeiro símbolo. Assim, se a palavra começa com “a” , o último símbolo deve ser diferente de “a”. No entanto, no meio da palavra podemos ter qualquer símbolo. O mesmo acontece com os símbolos iniciais sendo “b” ou “c”.

Questão 08

Linguagem: {w ∈ (0,1)* | w tem como prefixo 00 ou 111}

Diagrama de Estados:



Resultado das Computações:

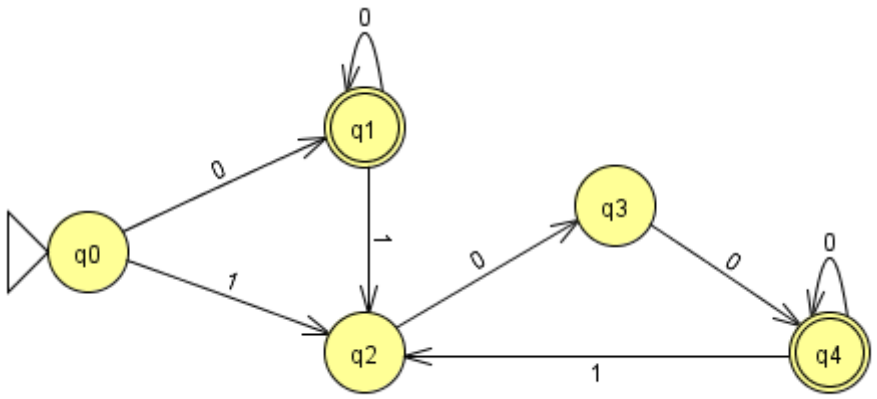
Input	Result
00001	Accept
11111	Accept
11100	Accept
0111	Reject
1100111	Reject
0100111	Reject

Se a palavra iniciar com “0” então a restrição de haver prefixo “00” deve ser atendida, solicitando logo em seguida outro 0. Os próximos símbolos podem ser qualquer um, indicado pelo laço no estado final. Se a palavra iniciar com “1” então a restrição de haver prefixo “111” deve ser atendida, solicitando logo em seguida duas leituras de outros 1’s. Novamente, os símbolos seguintes podem ser qualquer um dos disponíveis pelo alfabeto, indicado pelo laço.

Questão 09

Linguagem: **{w ∈ (0,1)* | w tem no mínimo dois 0's consecutivos após cada 1}**

Diagrama de Estados:



Resultado das Computações:

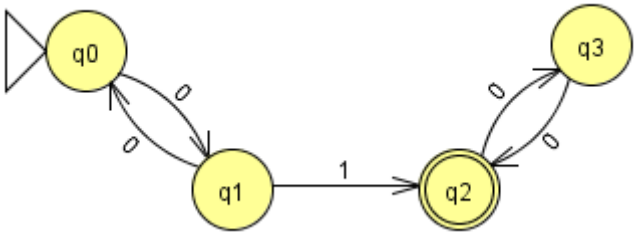
Input	Result
00000	Accept
100100	Accept
0100	Accept
0111	Reject
1100111	Reject
100100101	Reject

Aqui, onde houver um “1” a direção seguinte são os estados q2,q3,q4, os quais garantem dois “0s” consecutivos após cada “1”.

Questão 10

Linguagem: {w ∈ {0,1}* | w possui um único 1 e um número ímpar de 0's antes deste 1 e um número par de 0's após este 1}

Diagrama de Estados:



Resultado das Computações:

Input	Result
01	Accept
0100	Accept
000100	Accept
010	Reject
100	Reject
001	Reject

Primeiramente, para garantir um número ímpar de “0s” antes do “1” , foi criado um ciclo a partir do estado inicial q0 com q1. Para garantir o número par de “0s” depois do “1”, novamente foi feito um ciclo mas este, por sua vez, permite nenhum “0” ou um número par de “0s” maior que zero. Este ciclo ocorre entre os estados q2 e q3.

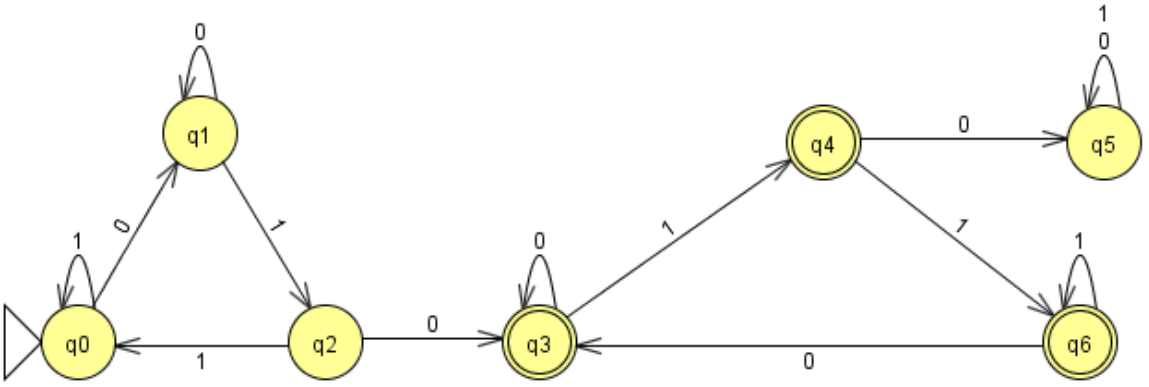
Questão 11

Três linguagens regulares quaisquer, **definidas por você**.

Questão 11.1

Linguagem: {w ∈ {0,1}* | w contém exatamente uma ocorrência da subpalavra 010}

Diagrama de Estados:



Resultado das Computações:

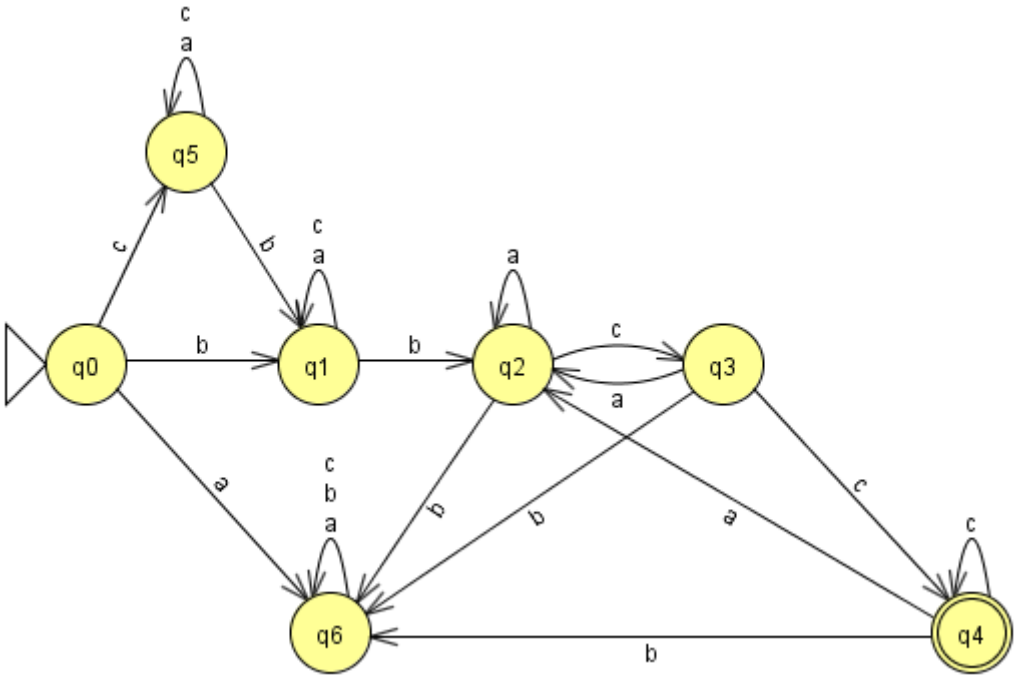
Input	Result
11010	Accept
1010011	Accept
01110010	Accept
10001110	Reject
100	Reject
111	Reject

Primeiramente, criamos 4 estados: q0, q1, q2, q3 e seguimos as transições com 010, como pedido, portanto então, q3 é final. Mas estas não são as únicas restrições. O laço em q0 “1” diz que podemos colocar quantos 1’s quisermos no início da palavra, mas não será final. O laço em q1 com leituras de quantos 0’s quisermos. Se em q2 a leitura for um “1”, então temos que voltar para o estado inicial, pois perdemos a condição de alcançar 010. Agora, teremos as transições com “0” e “1” em q3. Lendo 0’s podemos fazer um laço para q3. Caso haja a leitura de um “1”, não podemos continuar em q3, pois haveria casos de existir mais de uma leitura de 010. Então criamos o estado q4, que também será final. Se houver a leitura de 0 em q4 a transição levará para q5, o qual é um estado de erro. Se houver leitura de 1 em q4, vamos para q6. Há então o laço em q6 (caso de leitura de 1’s) e a transição com leitura de 0 voltando a q3.

Questão 11.2

Linguagem: {w ∈ {a,b,c}* | w não começa com a, contém exatamente dois b's e termina com cc}

Diagrama de Estados:



Resultado das Computações:

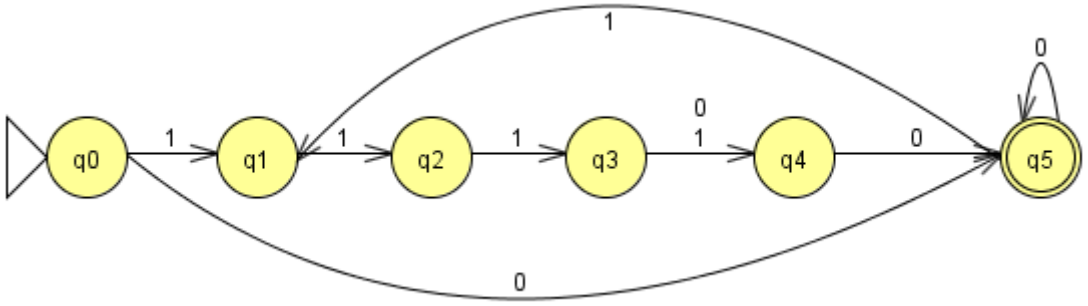
Input	Result
bbccc	Accept
caabacbaccc	Accept
bacbcaaccc	Accept
bacacc	Reject
ababcc	Reject
bbcca	Reject

Aqui temos q6 como estado de erro, pois não queremos começar com "a". Dos estados q0 a q4 seguindo as transições temos os dois b's e o sufixo cc, como requerido. Então, q4 é final. As outras restrições são: podemos também começar com "c", então passamos de q0 para q5 e de q5 para q1 com uma leitura de "b" para voltarmos a encontrar este primeiro b. Note os laços possíveis em q5, q1 e q2. Se houver mais leituras de b's nos estado q2, q3 ou a4 teremos erro, então essas transições são representadas para q6. Por fim, se ocorrer uma leitura de "a" no estado q4, voltamos para o estado q2.

Questão 11.3

Linguagem: {w ∈ (0,1)* | cada 1 é seguido de dois 1's e o segundo símbolo após a sequência de 111 é sempre 0}

Diagrama de Estados:



Resultado das Computações:

Input	Result
00	Accept
1110011110	Accept
11110	Accept
0100	Reject
11100001	Reject
1011	Reject

Aqui a estratégia foi: dada a leitura de um "1", precisamos de mais dois 1's seguidos. Essas leituras ocorrem das transições de q0 até q3. O símbolo seguinte pode ser qualquer um, mas o seguinte a este é obrigatoriamente "0". Por fim, temos o estado q5 representado como estado final contendo um laço de 0's caso haja tal leitura.

Considerações Finais

Com este trabalho, foi possível revisar os conceitos estudados na disciplina sobre os autômatos, bem como aprender a utilizar uma ferramenta para a construção dos mesmos. A ferramenta é bem intuitiva, o que facilitou a construção e compreensão dos autômatos de cada questão. Além disso, foi legal construir tais máquinas e deixá-las de um modo "bonitinho".

Referências

[1] VIEIRA, Newton José. **Introdução aos Fundamentos da Computação:** linguagens e máquinas. São Paulo: Pioneira Thomson Learning, 2006.