**Henrique de S. Q. dos Santos, NUSP 10819029**

**Witor M. A. de Oliveira, NUSP 10692190**

**Segunda avaliação**

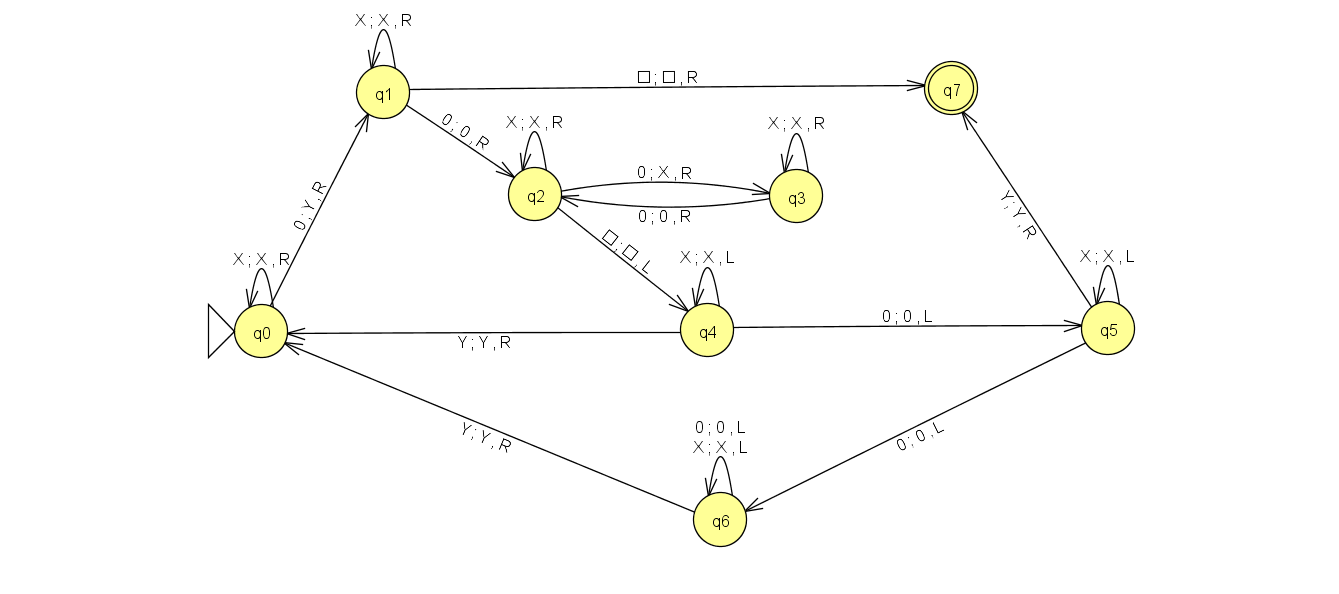
**1 -**

A máquina de turing que reconhece cadeias de 0, com tamanho , com , é a seguinte:

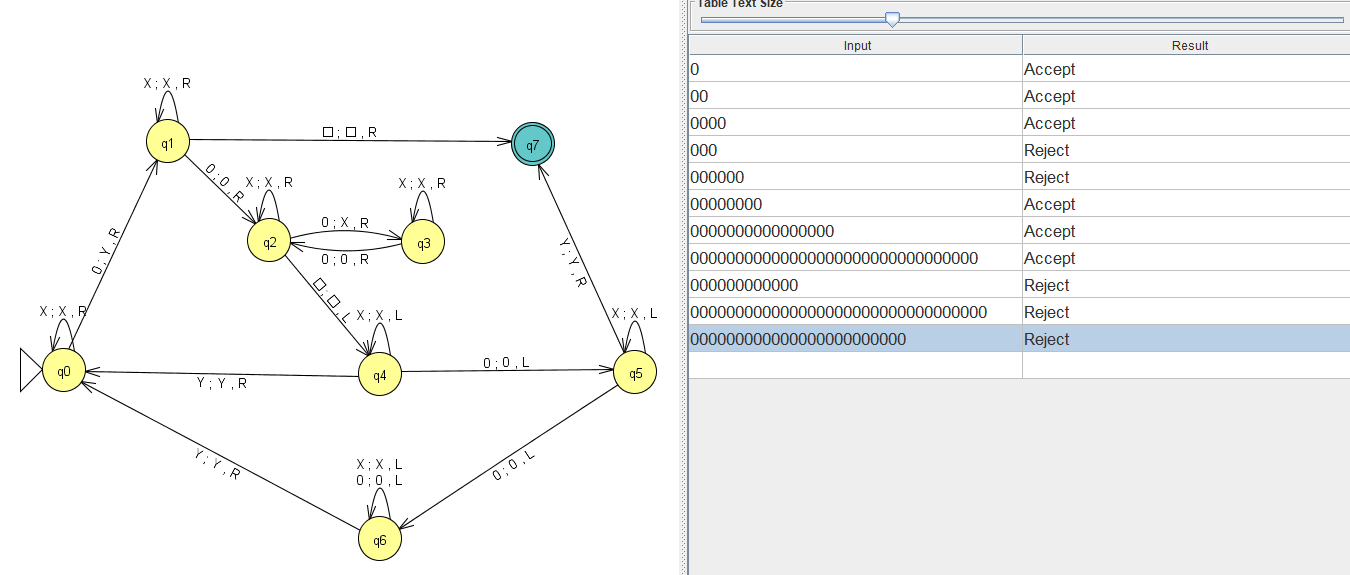
Onde:

* é o conjunto de estados;
* é o conjunto de símbolos aceitos pela máquina na entrada;
* é o conjunto de símbolos que a máquina pode escrever na fita;
* é o conjunto de funções de transição de estados e está definido na imagem abaixo, qualquer transição para um dado símbolo de entrada que não estiver representada na imagem, significa uma transição para um estado de erro;
* é o estado inicial;
* , símbolos em branco do máquina;
* é o conjunto de estados finais;

Diagrama de estados:

****

Resultados:



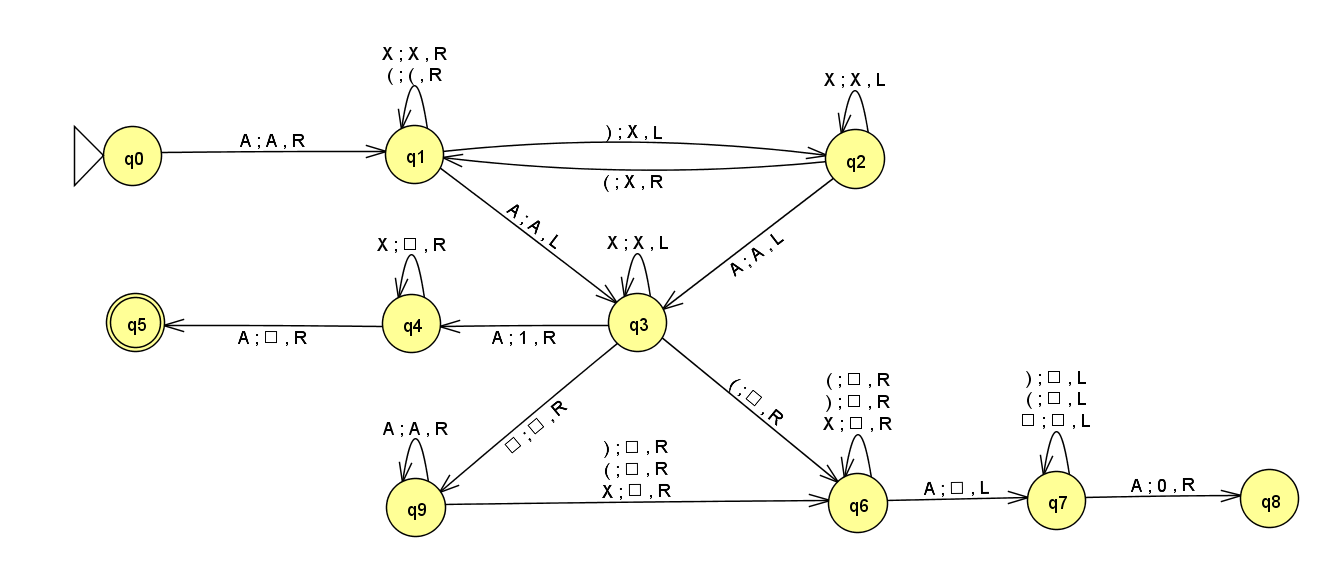
**2 -**

A máquina de turing que verifica se uma sequência de parênteses está bem formatada é a seguinte:

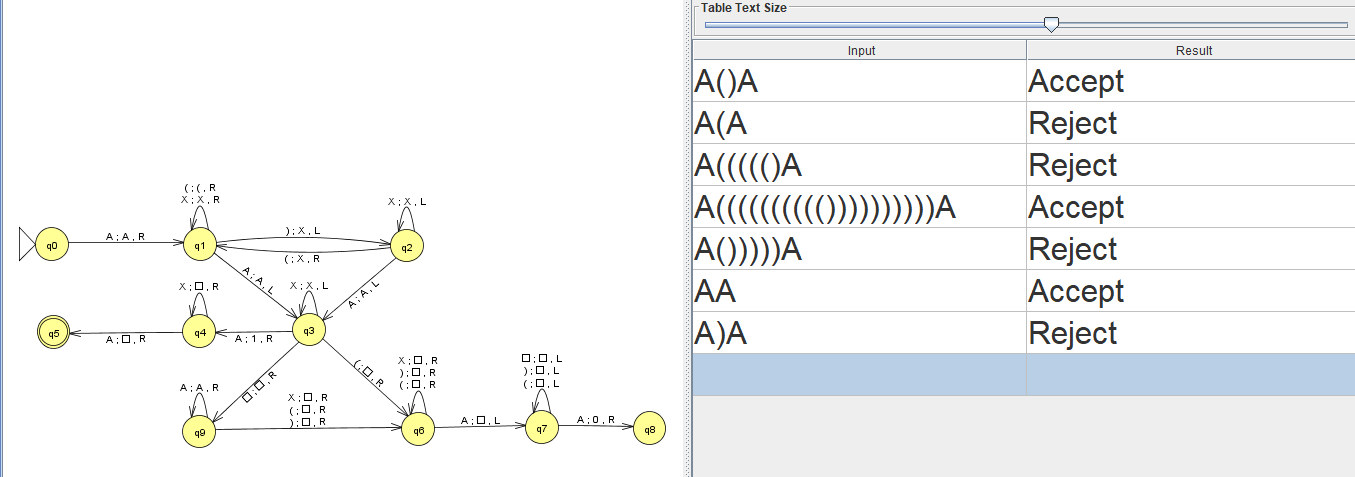
Onde:

* é o conjunto de estados;
* é o conjunto de símbolos aceitos pela máquina na entrada;
* é o conjunto de símbolos que a máquina pode escrever na fita;
* é o conjunto de funções de transição de estados e está definido na imagem abaixo, qualquer transição para um dado símbolo de entrada que não estiver representada na imagem, significa uma transição para um estado de erro;
* é o estado inicial;
* , símbolos em branco do máquina;
* é o conjunto de estados finais, ainda que nesse caso não importa se parou em um estado final ou não, visto que a máquina escreve na fita a resposta correspondente a computação da função;

Diagrama de estados:

****

Resultados:

****

**3 -**

O algoritmo utilizado para construir a máquina de turing que realiza a subtração *monus* de dois números inteiros () representados por zeros seguidos de um símbolo 1 e zeros, para e quando , = 0, é o seguinte:

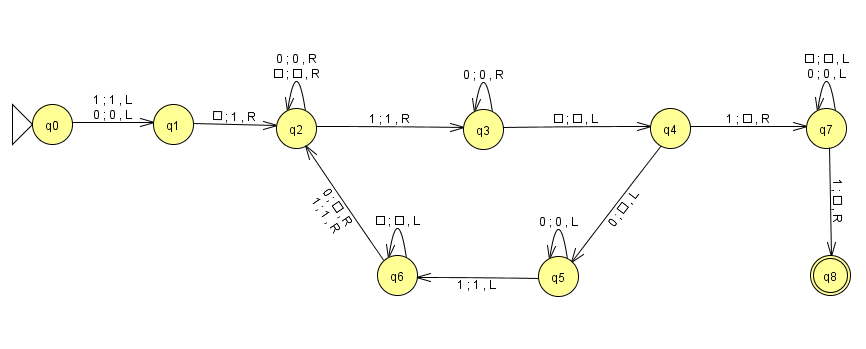
1. Iniciar escrevendo um símbolo 1 a esquerda do primeiro 0 da cadeia de entrada;
2. Ler todos os caracteres da esquerda para a direita até encontrar o primeiro caractere branco depois do caractere 1 que separa os dois números;
3. Depois ler da direita pra esquerda até encontrar o primeiro 0 antes do caractere 1 que separa os dois números, caso ele exista, substituir ele por um caracter em branco e ir para o passo 4, caso contrário o caracter 1 é sobrescrito por um caractere branco, é feita a leitura de todos os caracteres até chegar ao caractere 1 colocado no início da cadeia, ele é sobrescrito com o caractere branco e o algoritmo se encerra;
4. Ler todos os símbolos da direita pra esquerda até se chegar no primeiro caractere 0 depois do caractere 1. Quando isso acontece o símbolo 0 encontrado é sobrescrito com um caractere branco;
5. Repetir o passo 2;

Sendo definida assim essa máquina de turing

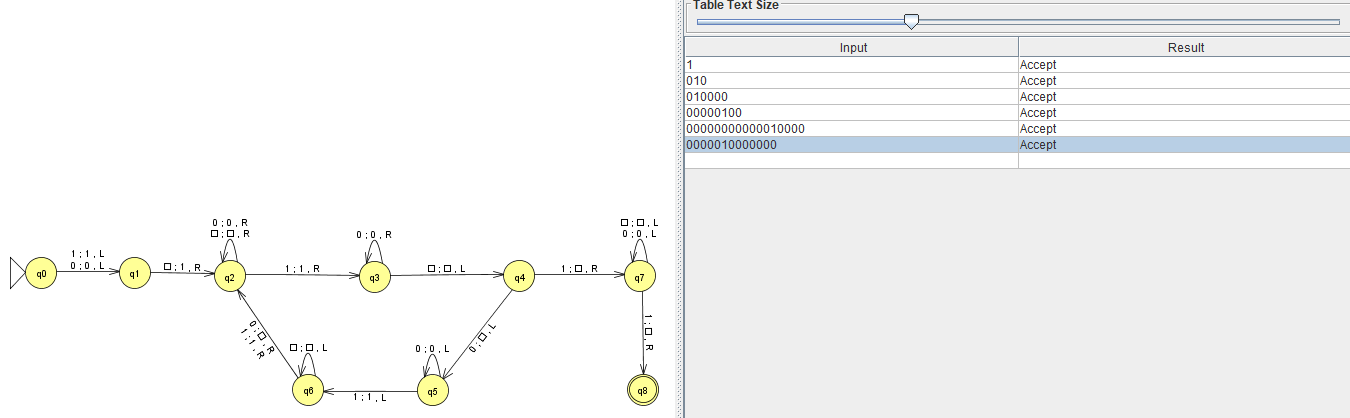
Onde:

* é o conjunto de estados;
* é o conjunto de símbolos aceitos pela máquina na entrada;
* é o conjunto de símbolos que a máquina pode escrever na fita;
* é o conjunto de funções de transição de estados e está definido na imagem abaixo, qualquer transição para um dado símbolo de entrada que não estiver representada na imagem, significa uma transição para um estado de erro;
* é o estado inicial;
* , símbolos em branco do máquina;
* é o conjunto de estados finais, ainda que nesse caso não importa se parou em um estado final ou não, visto que a máquina escreve na fita a resposta correspondente a computação da função;

Diagrama de estados:



Resultados:

****

**4) Forneça Autômatos com Pilha determinísticos para aceitar as linguagens a seguir:**

O **ACP** é definido por uma sétupla , onde

é o conjunto finito de estados;

é o alfabeto da cadeia de entrada;

é o alfabeto da pilha;

é a função de transição;

é o estado inicial;

é o símbolo inicial da pilha;

é o conjunto de estados finais.

Ele é determinístico se

1. Para cada e , se é não vazio, então é vazio ;
2. contém um só elemento , .

**OBS:** Na letra **a)** e **b)**, nós consideramos que, se o autômato receber uma cadeia 11111101, ele iria aceitar, pois ignoraria a cadeia ‘111111’ e aceitaria o ‘01’ seguinte.

1. Seja o APD com **aceitação por estado final**, onde:

= { q0, q1, q2, q3 };

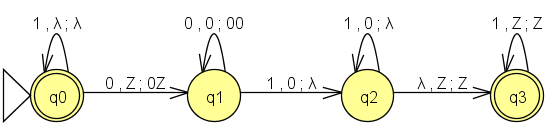
= { 0, 1 };

= { Z, 0 };

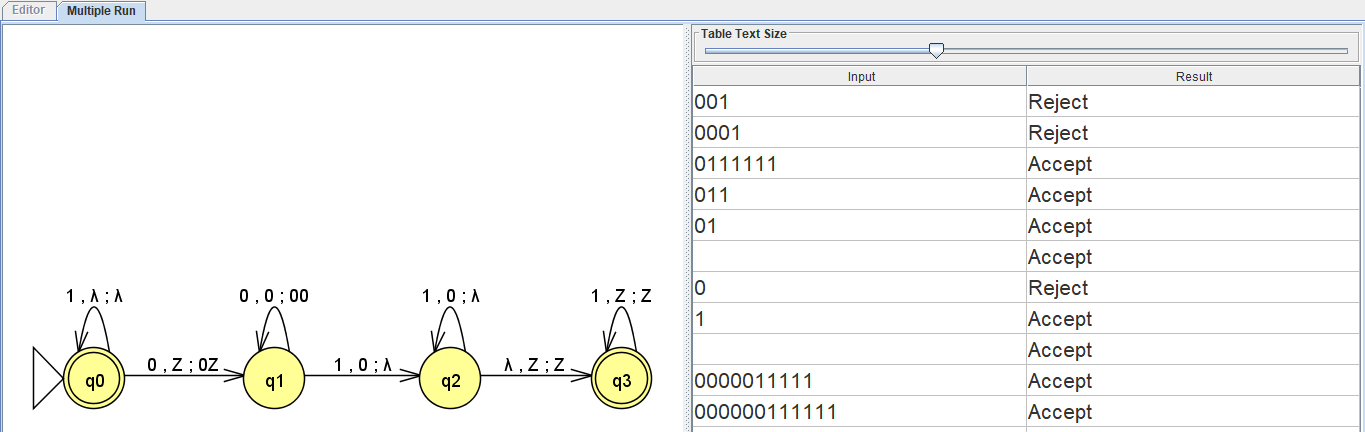
= { q0 };

= { Z };

= { q0, q3 }.



Resultados:



1. Seja o APD com **aceitação por** **estado final:**

= { q0, q1, q2 };

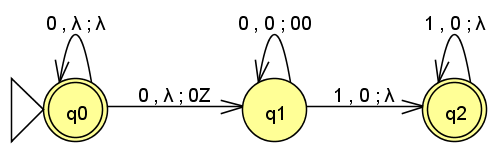
= { 0, 1 };

= { Z, 0 };

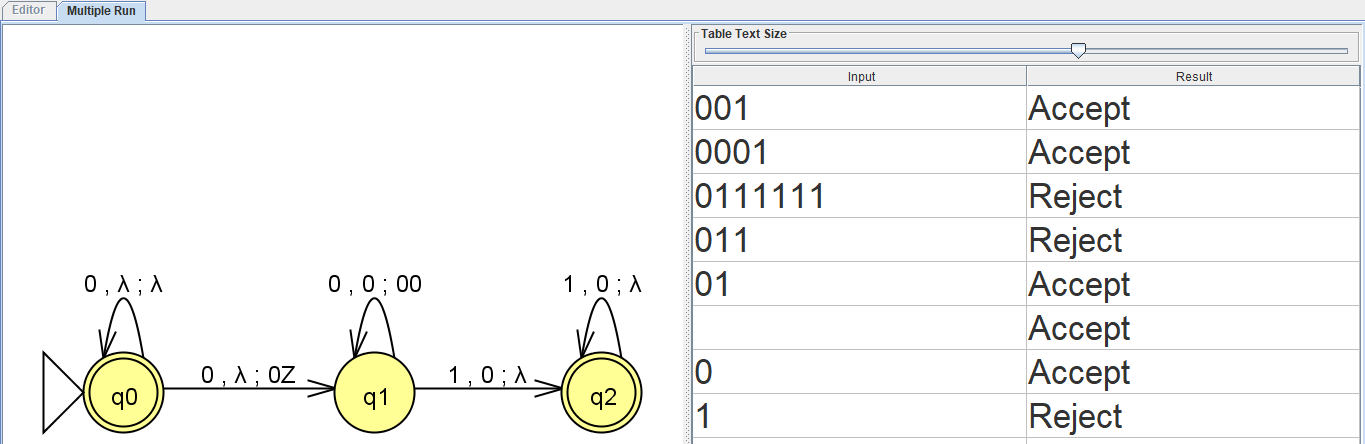
= { q0 };

= { Z };

= { q0, q2 }.



Resultados:



1. Seja o APD com **aceitação por estado final:**

= { q0, q1, q2, q3, q4, q5 };

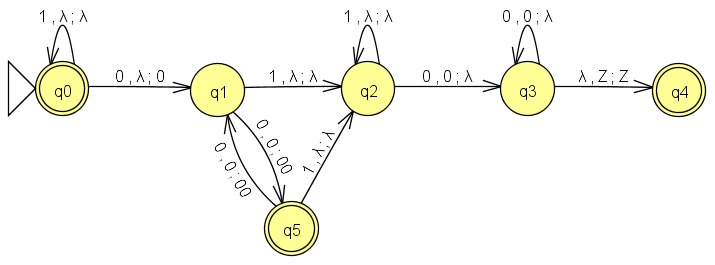
= { 0, 1 };

= { Z, 0 };

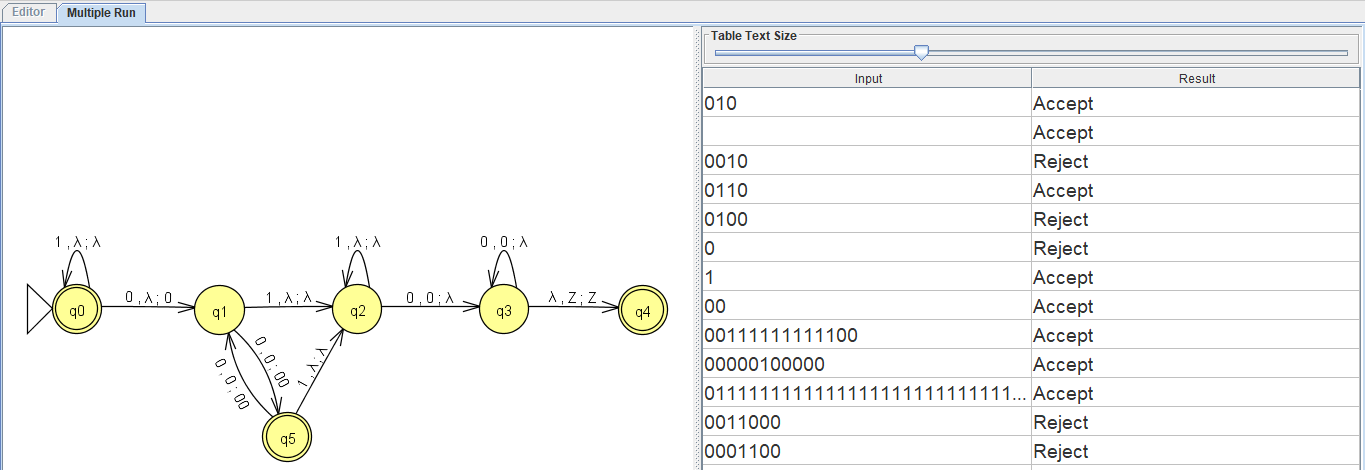
= { q0 };

= { Z };

= { q0, q4, q5 }.



Resultados:



**5 -**

Se uma linguagem é regular, ela pode ser gerada por uma gramática regular. Prova:

Se L é regular**,** então existe um autômato finito determinístico M = tal que . De M, podemos facilmente construir uma gramática regular que gera L, onde é definida por:

