Faculdade Cotemig

Introdução a Teoria da Computação

Trabalho Prático Implementação de APD

Belo Horizonte, 23/05/2014

Michael Douglas Rodrigues

Mateus Barros

Renato Renato Joncew

Rodrigo Alcântara

Conteúdo

1. Introdução

Este trabalho consiste em implementar um simulador de Autômato de Pilha(AP), onde o usuário poderá construir um autômato de pilha e simular seu funcionamento com uma palavra de entrada, A segunda etapa foi implementar dentro do simulador um modelo de APD com as seguintes definições:

Notação:

[] Significa 0 ou mais vezes

{} Significa 1 ou mais vezes

<> Significa "Opcionalmente"

"" Um texto constante

Gramática:

1) Alfabeto = {0,1,2,3,4,5,6,7,8,9, (,), +, -, *, /, a,...,z}
2) Variáveis auxiliares = {expression., constant, operator, digit, function, func1}
3) Variável de partida = expression.

Regras:

expression. : : = constant | (expression) | expression. Operator expression. | function

constant :: = [digit]<.{digit}>

digit ::= 0|1|2|3|4|5|6|7|8|9

function :: = func1(constant)

func1 :: = "abs"| "sqrt"| "sen"| "cos"| "tan"

operator :: = +|-|/|*

Definição de APD:

Um APD é um autômato com estados finitos e utiliza uma memória auxiliar em forma de pilha.

Autômatos com pilha diferem da definição normal de máquinas de estados finitos de duas maneiras:

Eles podem fazer uso da informação que está no topo da pilha para decidir qual transição deve ser efetuada;

Eles podem manipular a pilha ao efetuar uma transição.

Autômatos com pilha escolhem uma transição analisando o símbolo atual na cadeia de entrada, o estado atual e o topo da pilha. Autômatos com pilha adicionam a pilha como recurso auxiliar, deste modo, dado um símbolo da cadeia de entrada, o estado atual e um símbolo no topo da pilha, uma transição é selecionada.

1.1 Objetivos

O objetivo deste documento é descrever o trabalho, ambiente utilizado para desenvolvimento, linguagem escolhida, manual de utilização do programa, estruturas de dados utilizada, código fonte e testes realizados

2. Ambiente utilizado para desenvolvimento e testes

Os recursos necessários para a desenvolvimento do programa, incluem:

- Hardware – memória RAM de 2GB, HD com 320 GB, Core I3 2.3GHz,
- IDE de desenvolvimento – Netbeans 8.0, JDK 7 e JRE 7
- Software de apoio –Notepad++
- Linguagem de desenvolvimento – JAVA

3. Estruturas de dados utilizadas:

As estruturas utilizadas foram as seguintes:

- SimpleStack: Pilha para poder armazenar as variáveis que serão empilhadas e desempilhadas
- ArrayList: Classe default do JAVA para armazenar Transições e Estados.
- Iterator: Estrutura para percorrer listas filas e pilhas.
- List: O alfabeto de entrada e o alfabeto da pilha são armazenados em uma List.
- PriorityQueue: Priority Queue é uma fila de execução, foi utilizada para armazenar a palavra de entrada e para facilitar a leitura dos char.

4. Código fonte:

O código fonte foi estruturado na linguagem JAVA com conceitos de POO, foi utilizado estruturas de vetores para simular uma matriz dinâmica onde um array de estados guarda os estados criados e dentro de cada estado há um array de transições. As transições são criadas e guardada dentro do estado de origem configurado.

Q1	Q2	Q3	Q4	Q5	Q6
T1	T5	T7	T11	T13	T14
T2	T6	T8	T12		T15
T3		T9			T16
T4					T17
					T18

A Primeira linha representa o array de estados Q1, Q2, Q3... e as colunas representam o array de transição (T1, T2, T3,...), dentro de cada Estado. Ao configurar um estado, o primeiro sempre será inicial, e cabe ao usuário definir se este estado será final ou não. Os próximos estados não poderão ser iniciais, e é definido pelo usuário se ele será final ou não.

Para criar as transições, primeiramente deve se definir o alfabeto de entrada e o alfabeto da pilha e criar os respectivos estados de origem e destino.

Os símbolos do alfabeto de entrada poderá ser qualquer símbolo da tabela ASCII menos o símbolo "#" que foi escolhido para representar lambda no alfabeto da pilha. Lembrando que não pode haver símbolos repetidos e não pode haver símbolos do alfabeto da pilha no alfabeto de entrada, e vice e versa, exemplo:

Alfabeto de entrada: a, b.

Alfabeto da pilha: x, y, #.

Estados: Q1, Q2.

Após definir o alfabeto de entrada e o alfabeto da pilha e criar os estados de origem e destino, pode se configurar a função de transição da seguinte forma:

Símbolo: deve ser um símbolo do alfabeto de entrada.

Estado Origem: pode ser qualquer estado criado.

Estado Destino: pode ser qualquer estado criado.

Desempilha: deve ser um símbolo do alfabeto da pilha.

Empilha: deve ser um símbolo do alfabeto da pilha.

Exemplo de configuração de transição:

Símbolo: a

Estado Origem: Q1

Estado Destino: Q2

Desempilha: #

Empilha: x

Após configurar o autômato, o usuário poderá entrar com uma palavra e o programa irá verificar se está é reconhecida no autômato configurado.

Ao entrar com uma palavra no simulador o teste de reconhecimento acontecerá da seguinte forma:

O simulador primeiramente procurará por um estado final, após achar o estado final ele procurará pelo estado inicial para começar a ler a palavra passada.

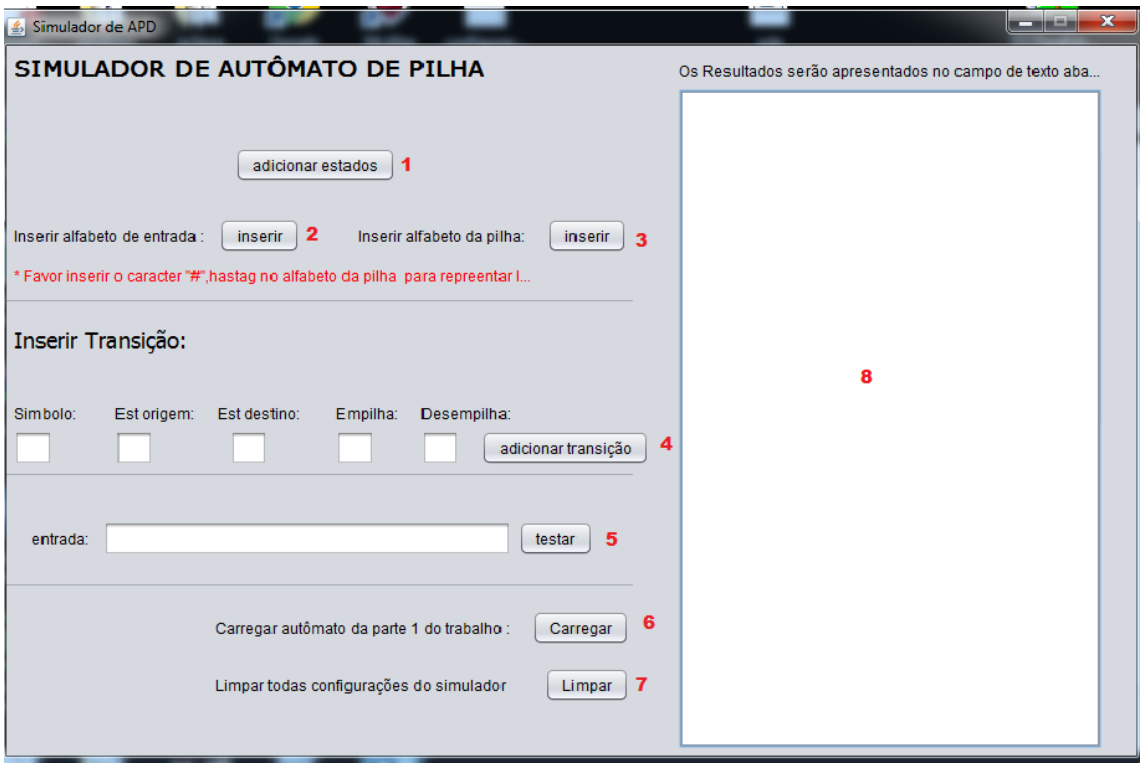
Após achar o estado inicial, ele verificará dentro do estado, no array de transições se há alguma transição que contém o símbolo lido pela palavra, se houver ele verifica nesta transição o símbolo de desempilhar e verifica se este símbolo é o topo da pilha ou é

lambda, após isso ele empilha o símbolo que tem para empilhar e chama o próximo estado. O mesmo procedimento acontece até o simulador ler a palavra por completo. Após ler a palavra ela será reconhecida se atender os seguintes critérios:

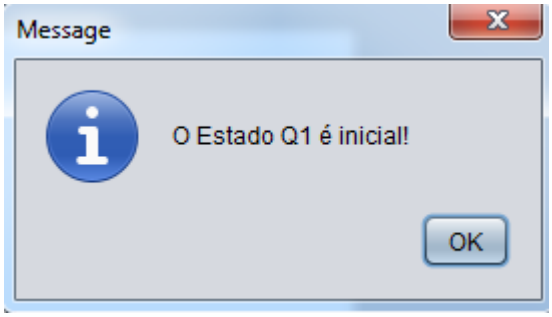
- Parar em um estado final
- A pilha estiver vazia
- A palavra for lida por completo

Caso um desses três critérios não forem atendidos será apresentada a mensagem "Não reconhecido!"

5. Manual de utilização do programa.

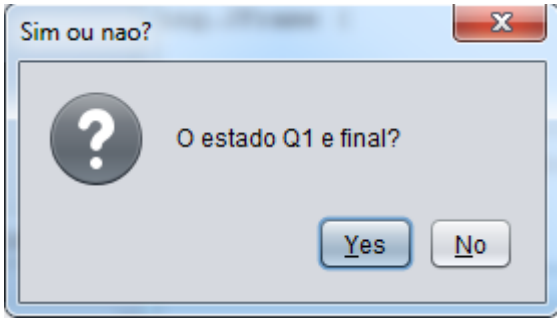


1 – **Botão para adicionar estados:** O primeiro estado adicionado sempre será inicial e cabe ao usuário escolher se ele será final ou não. Os estados adicionados são nomeados com a seguinte regra: concatenação da letra “Q” + índice.

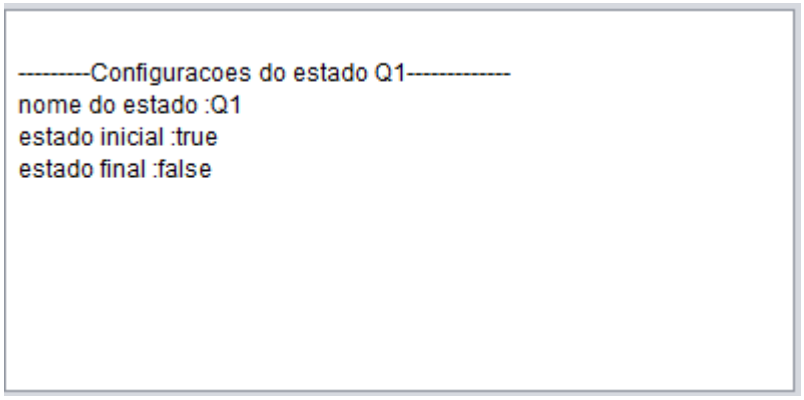


*obs. Os estados criados após o primeiro estado não serão iniciais.

Após clicar em OK o programa irá perguntar se o estado criado é final.



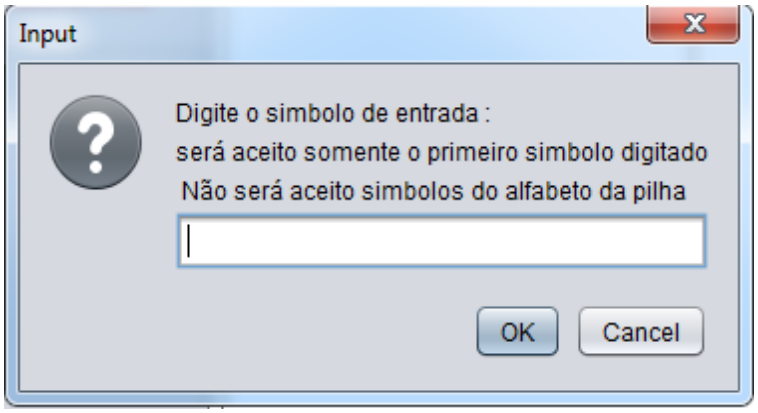
Após definir se o estado criado é final ou não, será apresentado na Jtext as configurações do estado criado:



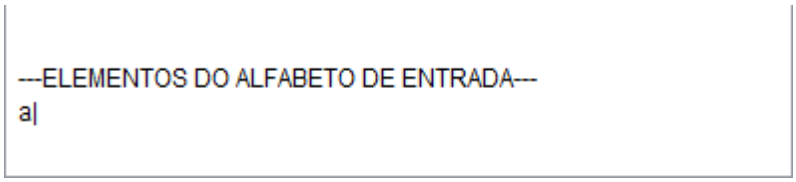
2 – Inserir símbolos do alfabeto de entrada: Para inserir os símbolos do alfabeto de entrada deve-se obedecer as seguintes regras:

- Será aceito somente o primeiro caractere digitado no campo

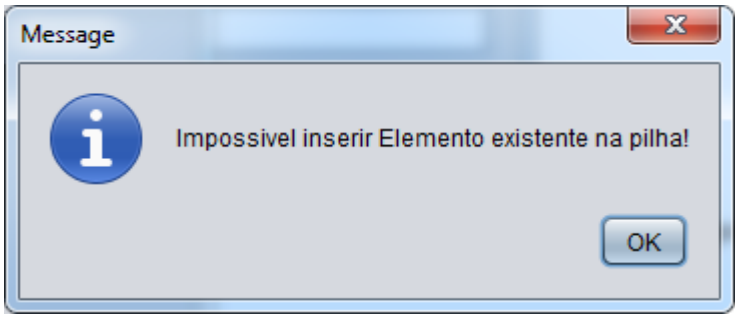
- Não será aceito caractere repetido
- Não será aceito caractere que pertencer ao alfabeto da pilha
- Não será aceito o caractere' #', pois o mesmo é utilizado para representar lambda no alfabeto da pilha.



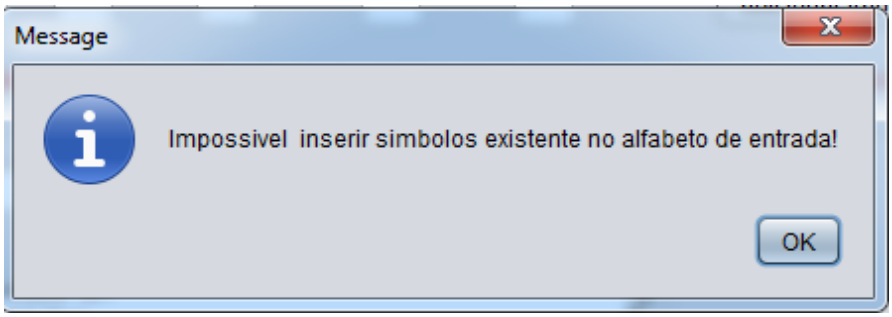
Após inserir o caractere desejado clique em ok, aparecerá no campo **9** o seguinte texto:



2.1 – Primeira exceção, inserir elemento existente no alfabeto da pilha:

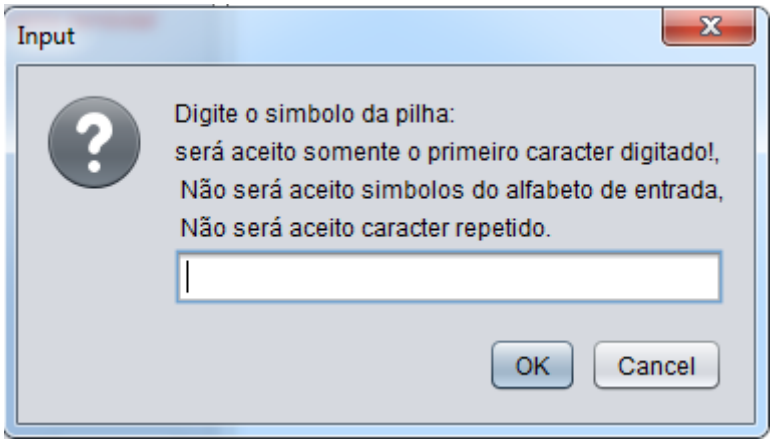


2.2 Segunda exceção, inserir elementos repetidos:



3 – Inserir elementos no alfabeto da pilha: Para inserir os símbolos do alfabeto da pilha deve-se obedecer as seguintes regras:

- Será aceito somente o primeiro caractere digitado no campo
- Não será aceito caractere repetido
- Não será aceito caractere que pertencer ao alfabeto da pilha
- Se quiser representar lambda, favor inserir o caractere' #'.



*obs. Não serão aceitos caracteres do alfabeto de entrada e não serão aceitos caracteres repetidos.

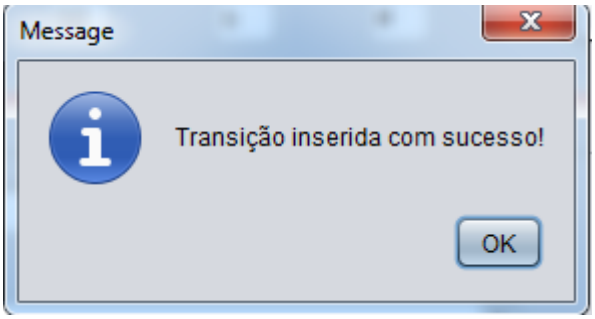
Após inserir o caractere desejado será exibido no campo **9** o seguinte texto:

4 – Adicionar transições

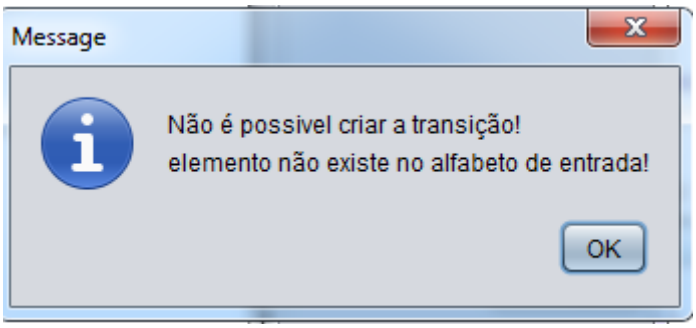
Para adicionar transições ao autômato, devemos informar o símbolo, o estado de origem, o estado destino, empilha e desempilha. Devem ser obedecidas as seguintes regras:

- Só serão aceitos no campo símbolo caracteres inseridos no alfabeto de entrada.
- Os estados de origem devem existir, e para ser inserido devem seguir o seguinte modelo: letra ‘Q’ maiúscula seguido do número índice, exemplo: Q1, Q2, Q3 Q4 etc.
- Só serão aceitos no campo empilha e desempilha caracteres inseridos no alfabeto da pilha, para inserir lambda use o caractere ‘#’, lembrando que o mesmo deve ser inserido no alfabeto da pilha.

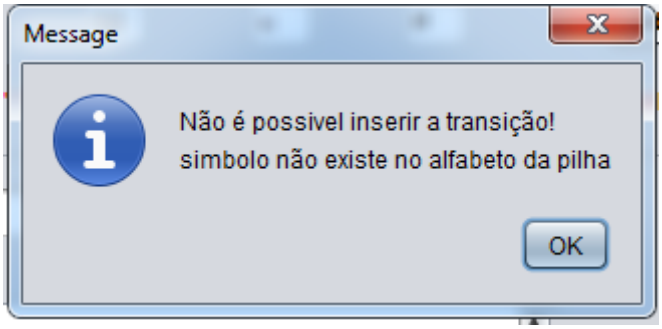
  Após obedecer as seguintes regras clique em adicionar transição, ela será inserida e aparecerá a seguinte mensagem:



4.1 – exceção 1- Inserir transição com símbolo que não pertence ao alfabeto de entrada:



4.1 exceção 2 – Inserir transição com empilhar e desempilhar que não pertence ao alfabeto da pilha:



4.2- demais exceções:

As demais exceções como inserir transição sem informar estados de origem ou destino configuram erro no simulador. Ao clicar em adicionar transição com alguma dessas exceções informadas acima, não acontecerá nada ao programa.

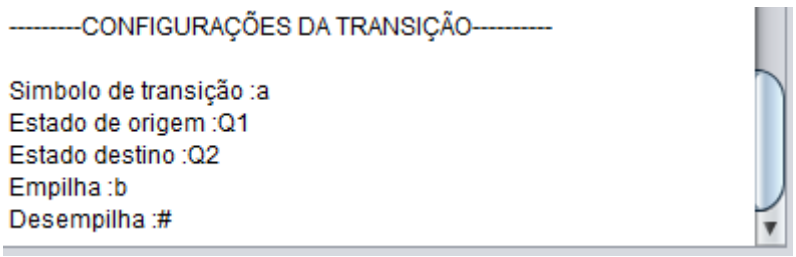
Ao inserir uma transição deverá aparecer o seguinte texto no campo 9.

4.3- Transições não determinísticas:

O simulador foi configurado para não aceitar transições não determinísticas. Acontece o não determinismo quando dentro dos autômatos há:

- Mais de um uma transição partindo do mesmo estado, com o mesmo símbolo e mesmo símbolo para desempilhar.
- Mais de uma transição partindo do mesmo estado, com o mesmo símbolo e um dos símbolos de desempilhar é lambda.
- Mais de uma transição do mesmo estado, com um dos símbolos sendo lambda.

Após configurar uma transição será apresentada no campo de texto as informações:

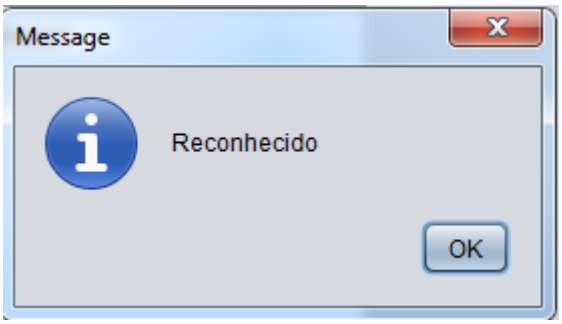


5- Campo de teste da entrada

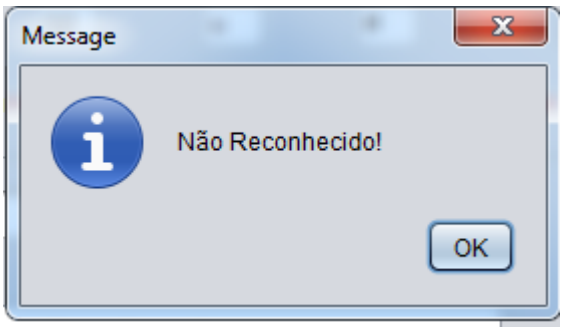
Insira no campo a palavra que deseja testar e clique em testar.

5.1 – Campo para testar a palavra informada no campo 5.

Se a palavra informada for executada e reconhecida, deverá aparecer a seguinte mensagem:



Se a palavra não for reconhecida, deverá aparecer a seguinte mensagem:



6- Transições não determinísticas:

Ao clicar em limpar todas as configurações do autômatos serão limpas e permitirá configurar um novo AP.

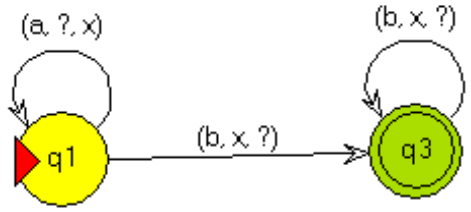
7 – Carregar

Ao clicar em carregar o programa irá carregar o autômato para reconhecer expressões matemáticas com parênteses, pedida na primeira parte do trabalho.

6. Testes Realizados:

Foram realizados diversos testes com modelos de autômatos de pilha, os resultados esperados foram obtidos com sucesso, segue a seguir modelo de teste criado e resultados esperados e obtidos:

Primeiro teste:



---------Configurações do estado Q1----------
Nome do estado :Q1
Estado inicial :true
Estado final :false

---------Configurações do estado Q2----------
Nome do estado :Q2
Estado inicial :false
Estado final :true

---ELEMENTOS DO ALFABETO DE ENTRADA---|
a|b|
---ELEMENTOS DA PILHA---
#-x-
---------CONFIGURAÇÕES DA TRANSIÇÃO----------

Símbolo de transição :a
Estado de origem :Q1
Estado destino :Q1
Empilha :x
Desempilha:#
---------CONFIGURAÇÕES DA TRANSIÇÃO----------

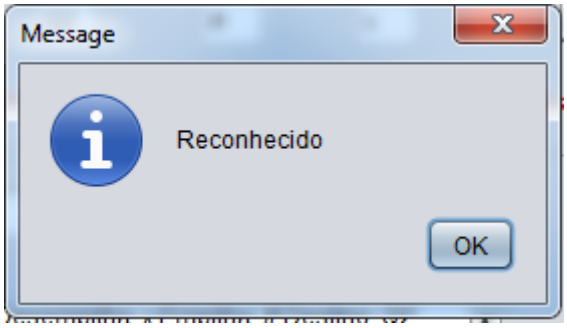
Símbolo de transição :b
Estado de origem :Q1
Estado destino :Q2
Empilha:#

Desempilha :x

---------CONFIGURAÇÕES DA TRANSIÇÃO----------

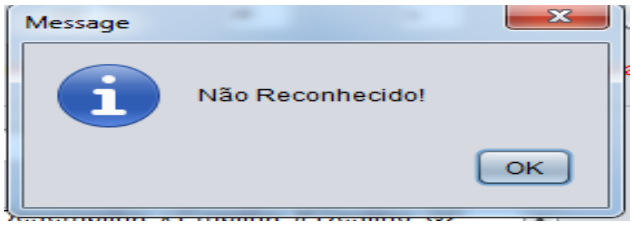
Símbolo de transição :b
Estado de origem :Q2
Estado destino :Q2
Empilha:#
Desempilha :x

Teste:
Palavra: aaabbb
Estado: Q1 Transição: a Desempilha: # Empilha: x Destino: Q1
Estado: Q1 Transição: a Desempilha: # Empilha: x Destino: Q1
Estado: Q1 Transição: a Desempilha: # Empilha: x Destino: Q1
Estado: Q1 Transição: b Desempilha: x Empilha: # Destino: Q2
Estado: Q2 Transição: b Desempilha: x Empilha: # Destino: Q2
Estado: Q2 Transição: b Desempilha: x Empilha: # Destino: Q2



Palavra: aaabb

Estado: Q1 Transição: a Desempilha: # Empilha: x Destino: Q1
Estado: Q1 Transição: a Desempilha: # Empilha: x Destino: Q1
Estado: Q1 Transição: a Desempilha: # Empilha: x Destino: Q1
Estado: Q1 Transição: b Desempilha: x Empilha: # Destino: Q2
Estado: Q2 Transição: b Desempilha: x Empilha: # Destino: Q2


Message
Não Reconhecido!
OK