

Artigo sobre Autômatos em JFLAP

Nome: Luiz Henrique Botega Beraldi

Professora: Cinthyan Renata Sachs Camerlengo de Barbosa

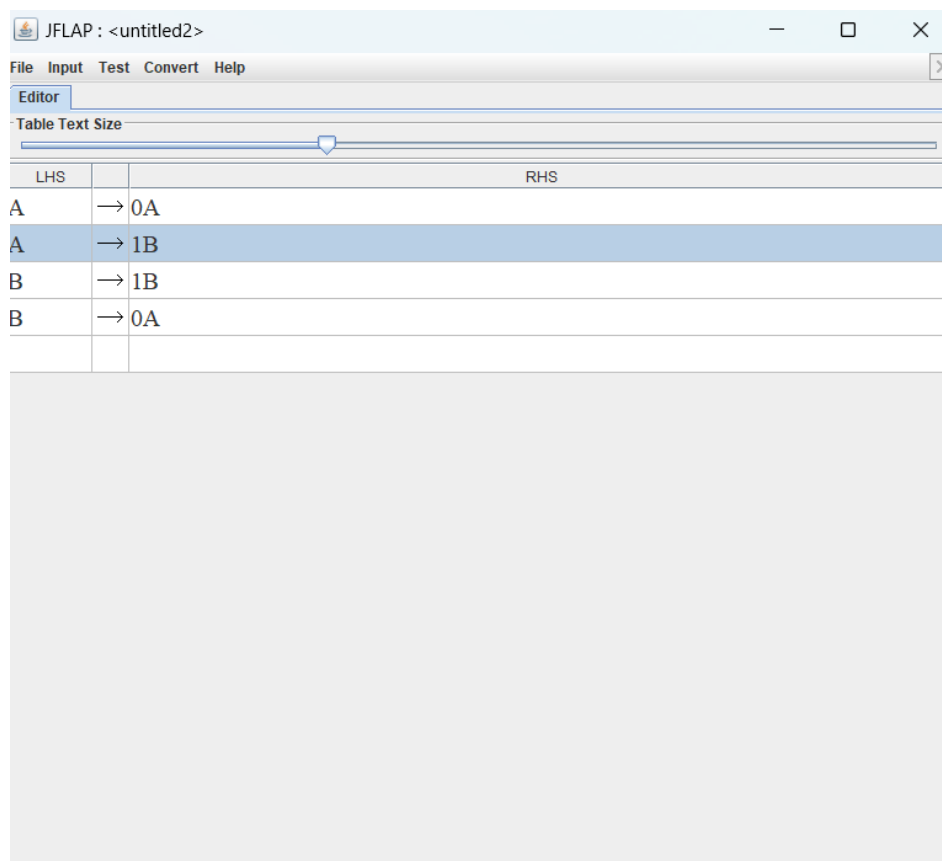
O intuito do artigo é comparar as resoluções e ajudas do JFLAP com os slides e aulas ministradas pela professora Cinthyan.

Slide 13

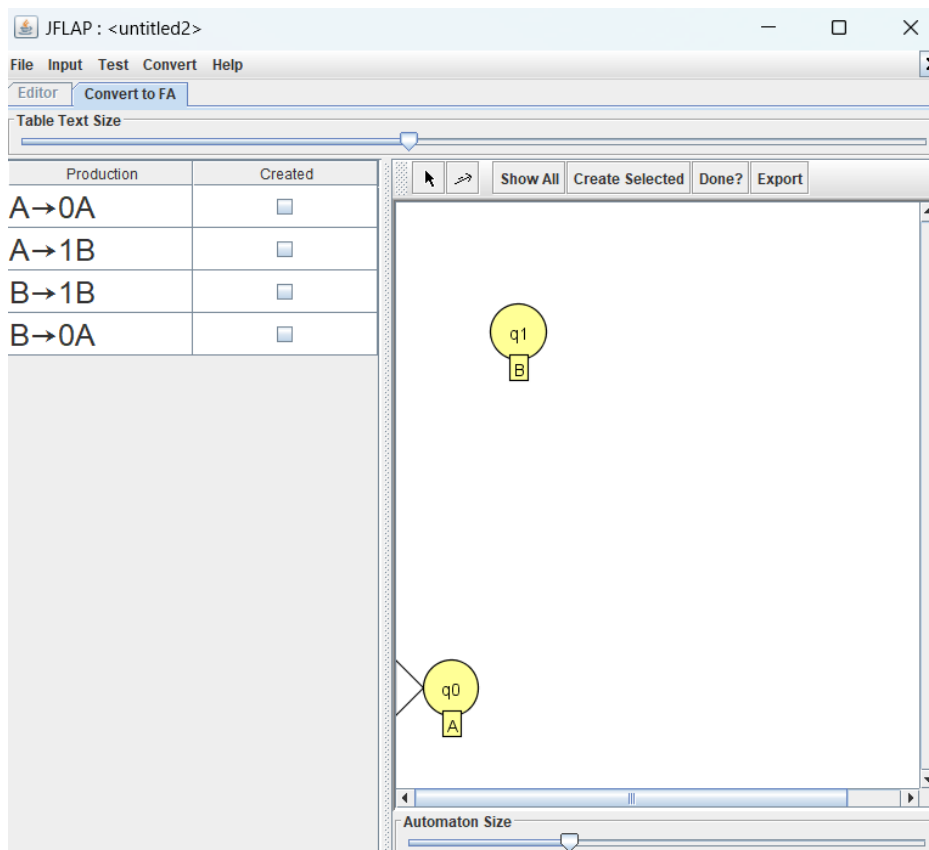
Exemplo 2.1

Transformar uma gramática linear à direita em um autômato finito

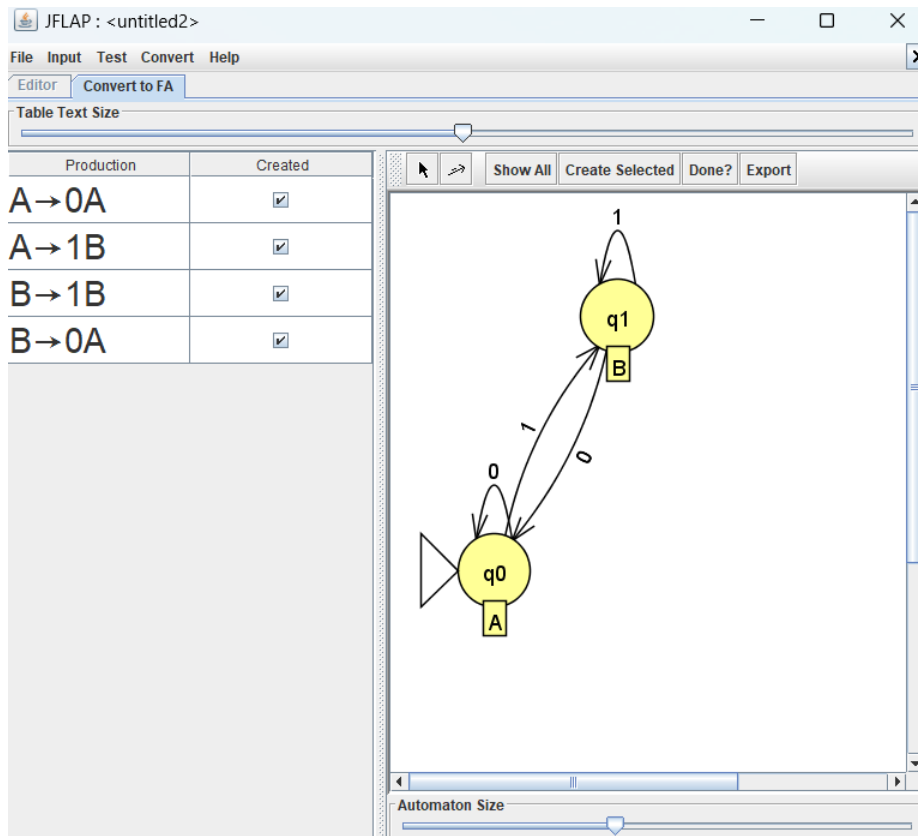
Indo em Grammar e colocando a gramatica do exemplo: $M = (\{A,B\}, \{0,1\}, \{(A,0) \rightarrow A, (A,1) \rightarrow B, (B,1) \rightarrow B, (B,0) \rightarrow A\}, A, \{B\})$



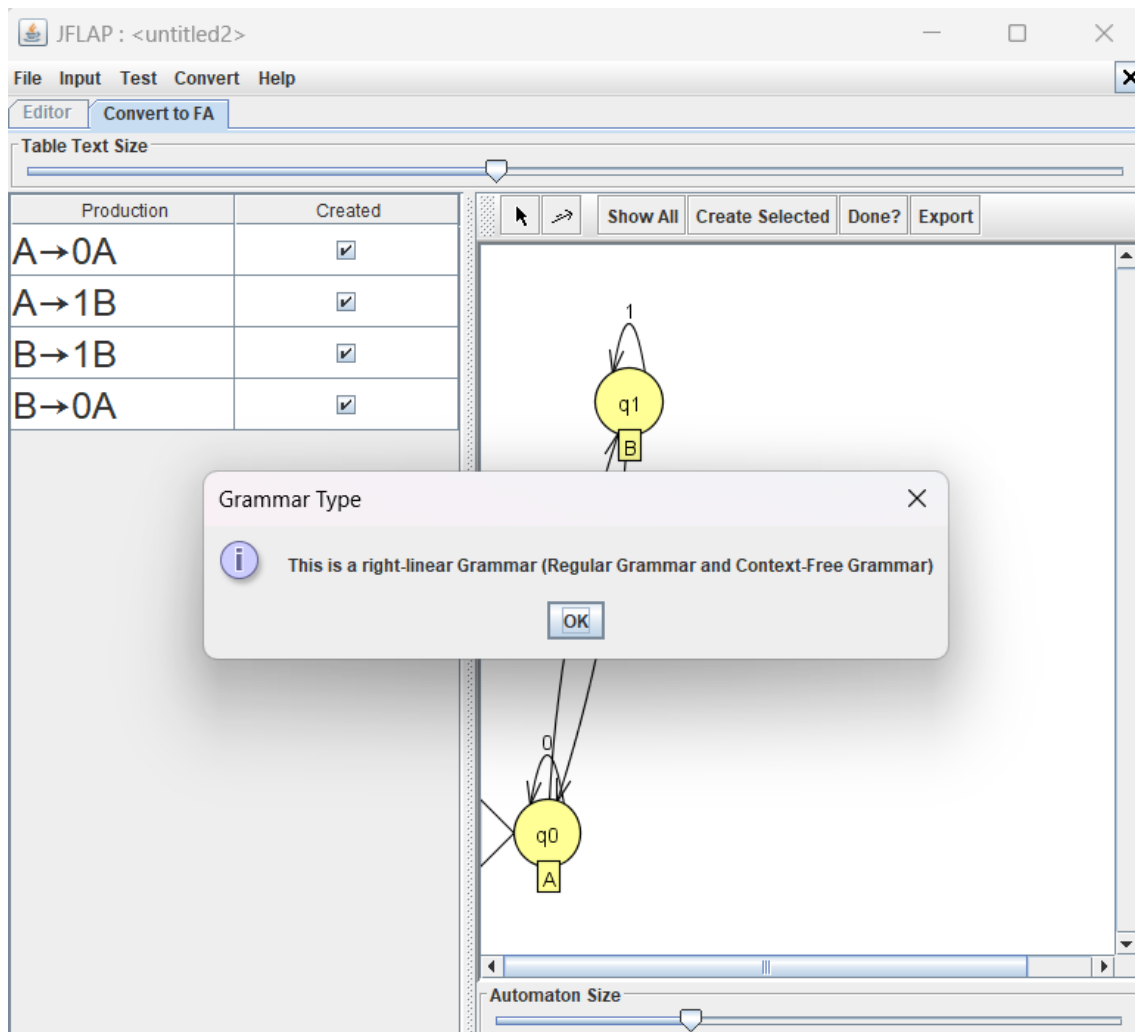
Agora clicando em Convert e depois em Convert Right-Linear Grammar to FA, aparecerá essa tela:



Clicando em Show All ele cria o autômato



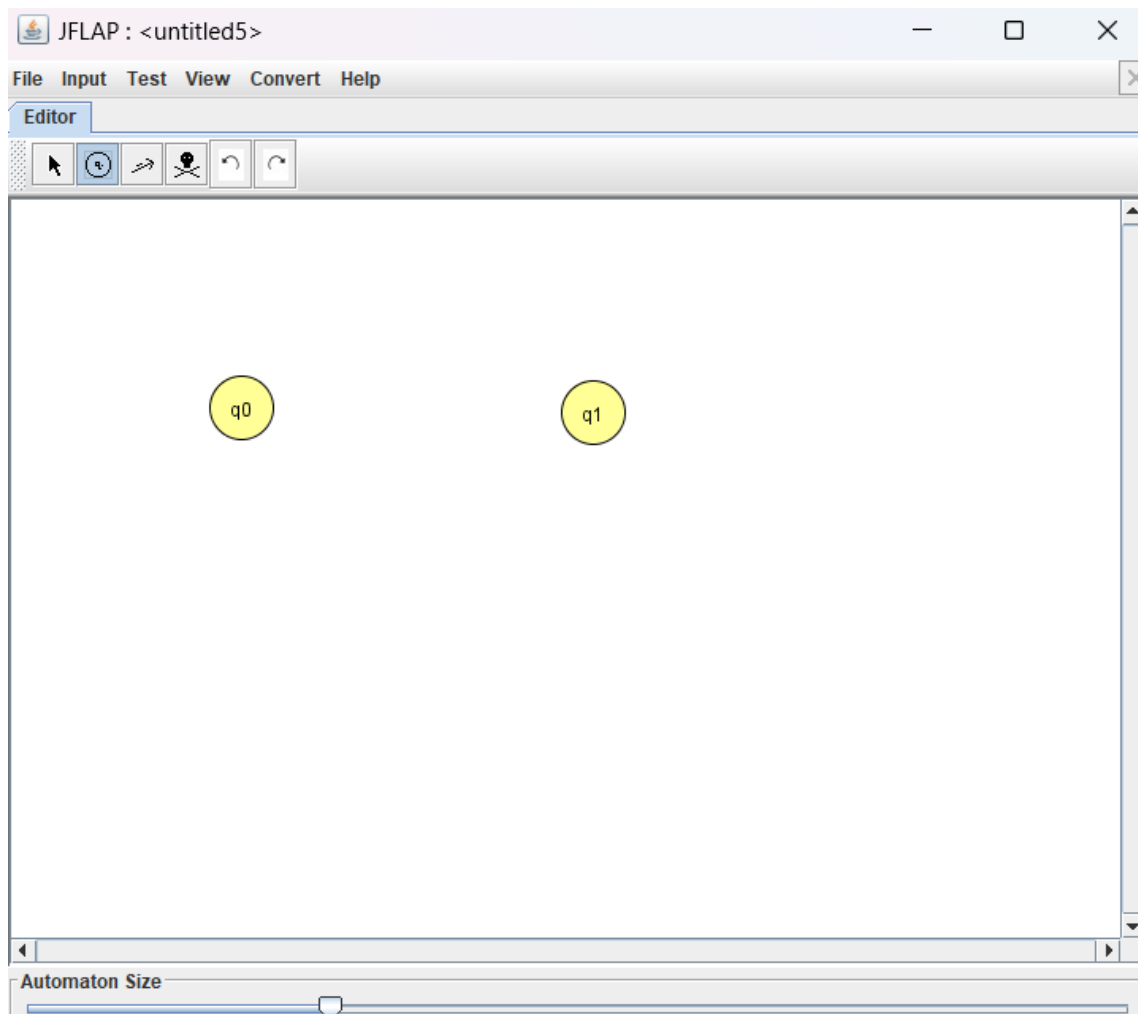
E clicando em Test e depois em Test For Grammar Type ele nos dá a certeza de que a gramática é linear à direita.



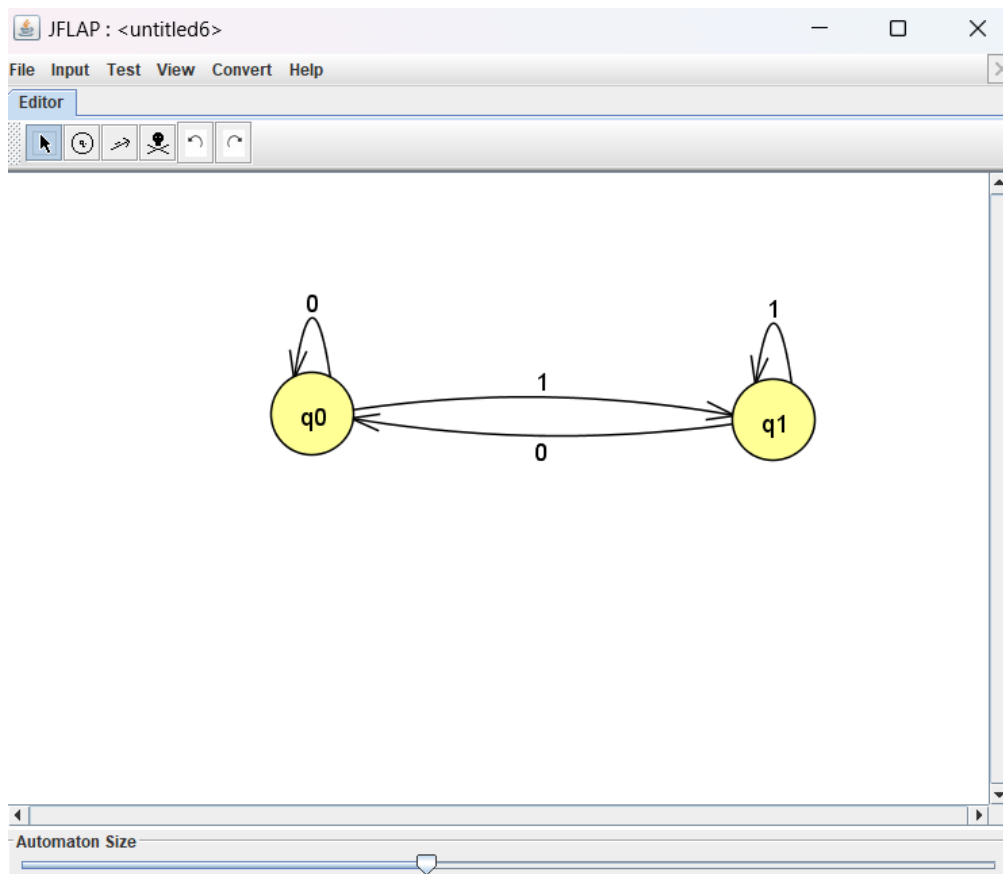
Exemplo 2.1

Transformar um autômato finito em uma gramática linear à direita

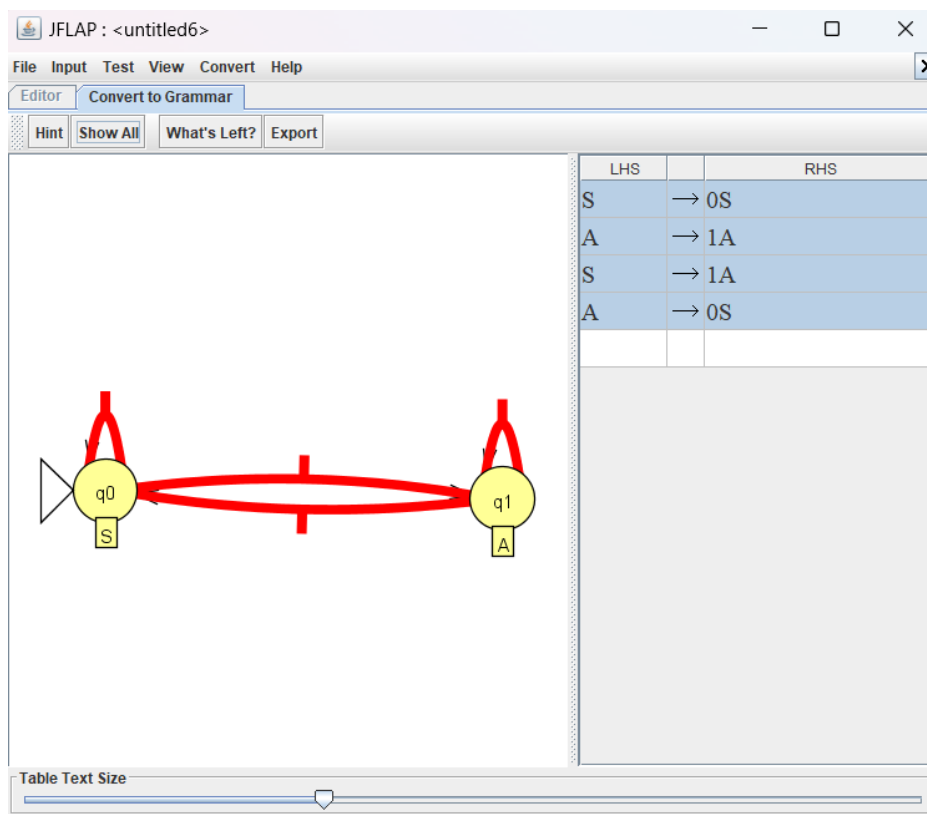
Indo em Finite Automaton e depois em State Creator podemos colocar nossos autômatos no JFLAP



E depois clicando em Transition Creator podemos colocar as transições dos autômatos



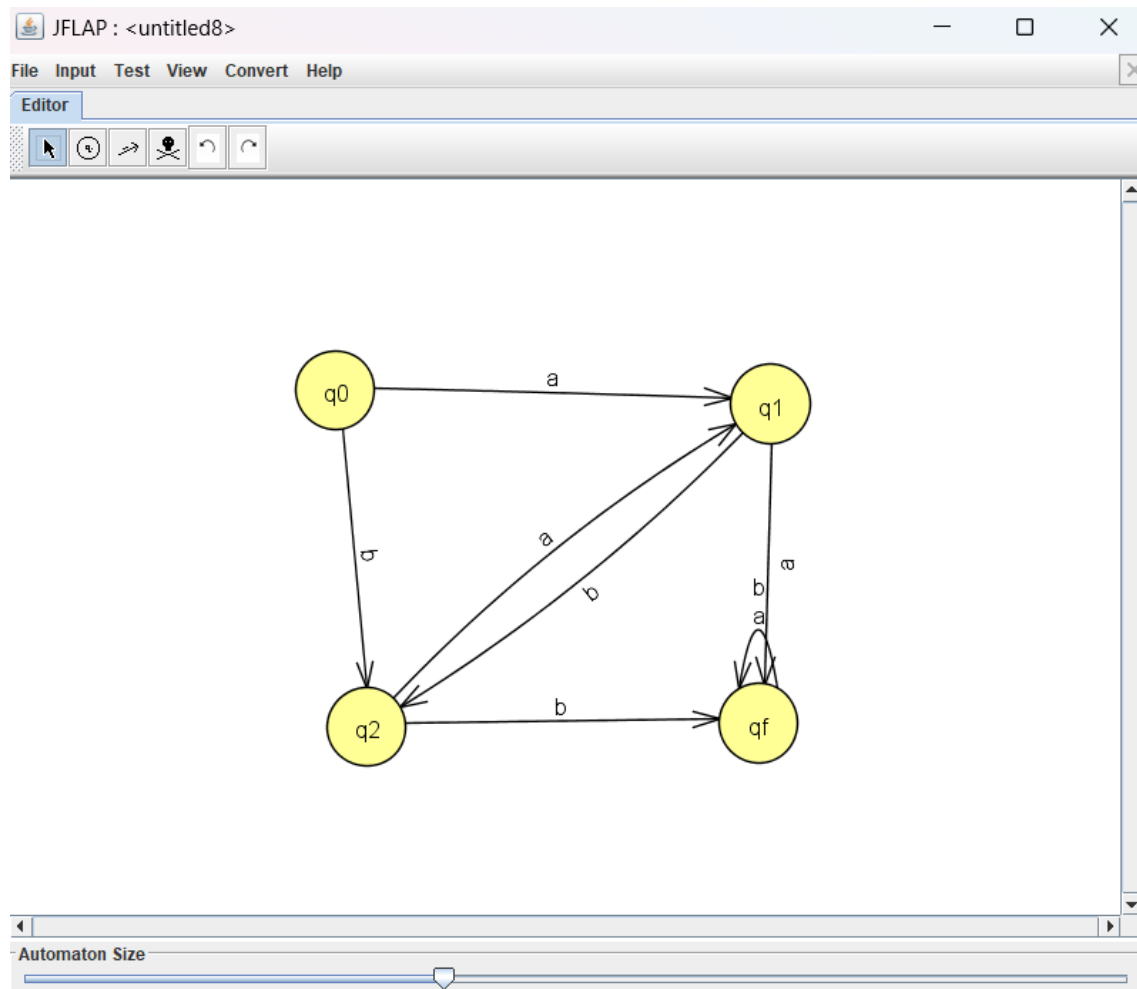
Agora clicando em Convert e depois em Convert to grammar e em Show All ele nos dá a gramática linear à direita:



Exemplo 2.2

Criar um autômato finito

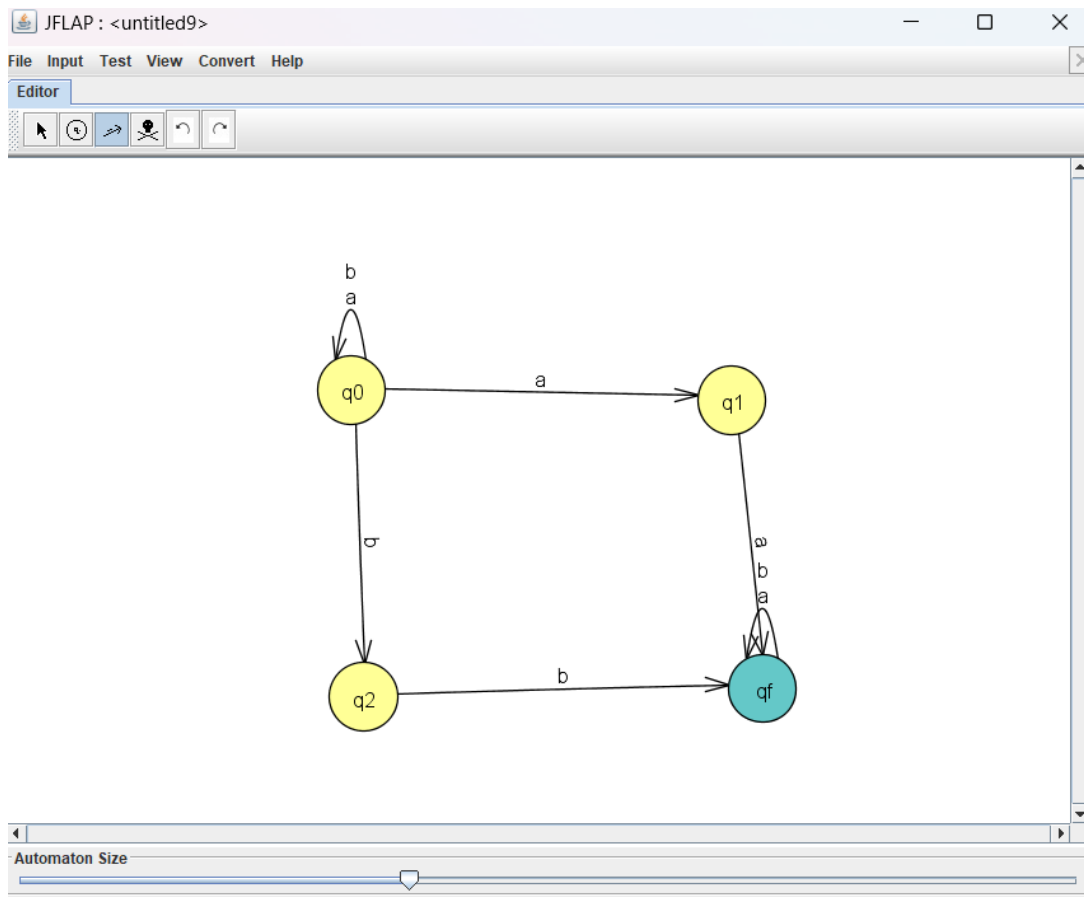
Clicando em Finite Automaton você pode criar seu próprio autômato, usando o autômato do exemplo 2.2, $M1 = (\{q_0, q_1, q_2, q_f\}, \{a, b\}, \delta, q_0, \{q_f\})$.



Exemplo 2.3

Testar uma ou várias palavras no autômato finito

Clicando em Finite Automaton colocando seu autômato e depois em Input e Multiple Run nós podemos testar palavras para saber se são ou não aceitas no autômato.



JFLAP : <untitled9>

File Input Test View Convert Help

Editor Multiple Run

```

graph LR
    start((( ))) --> q0(((q0)))
    q0 -- a --> q0
    q0 -- b --> q2((q2))
    q1((q1)) -- a --> qf(((qf)))
    q2 -- b --> qf
    qf -- a --> qf
    qf -- b --> qf
  
```

Table Text Size

Input	Result
aab	Accept
baaab	Accept

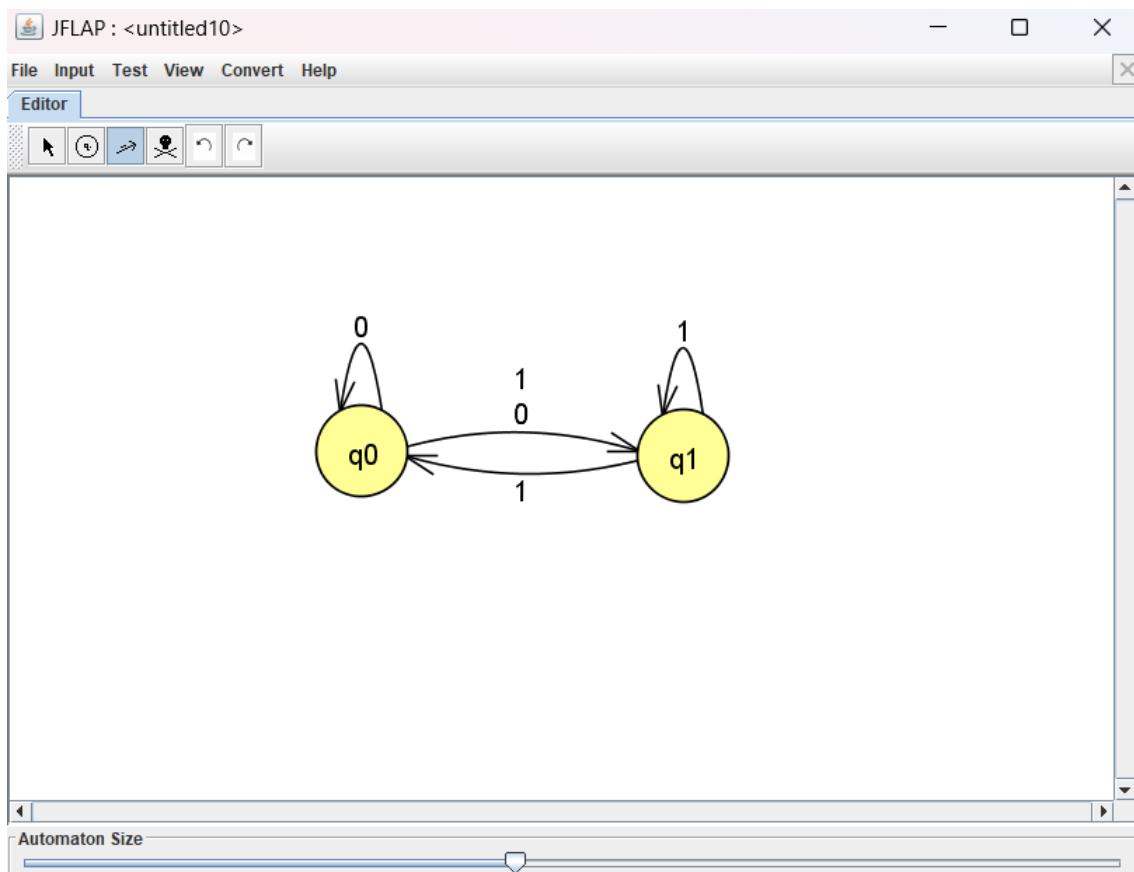
Load Inputs Run Inputs Clear Enter Lambda View Trace

Slide 15

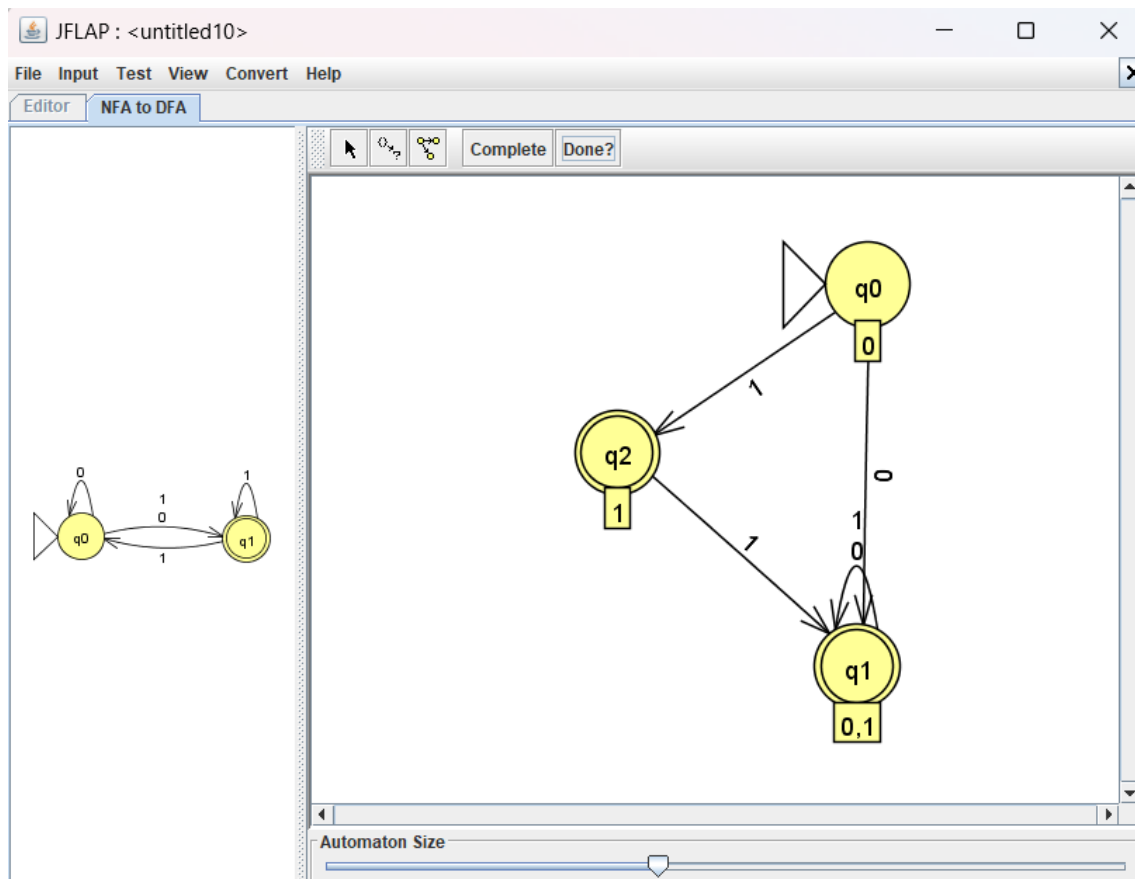
Exercício

Converter um autômato finito não determinístico em um autômato finito determinístico

Clicando em Finite Automaton e colocando o autômato do exercício ele fica assim:



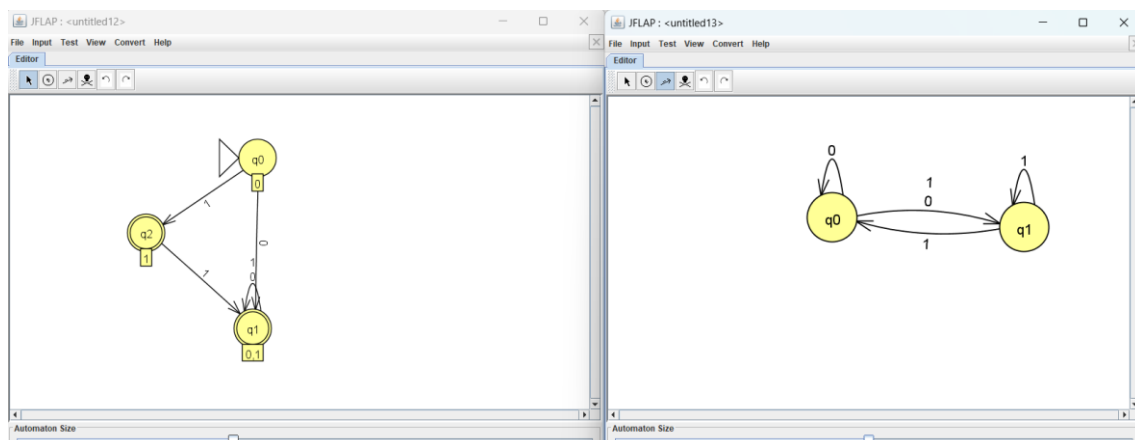
Agora clicando em Convert e depois em Convert to DFA e em Complete nós convertemos um AFND para um AFD:



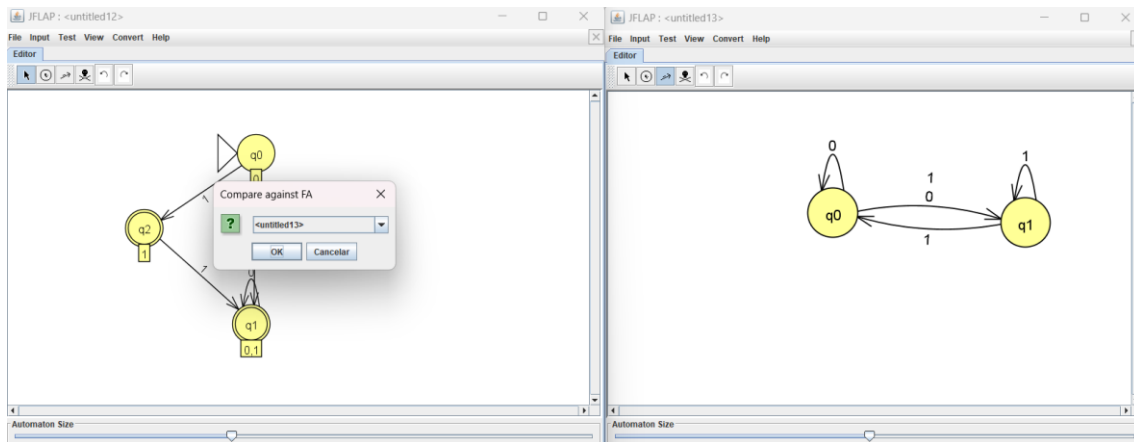
Exercício

Comparar equivalência entre autômatos finitos

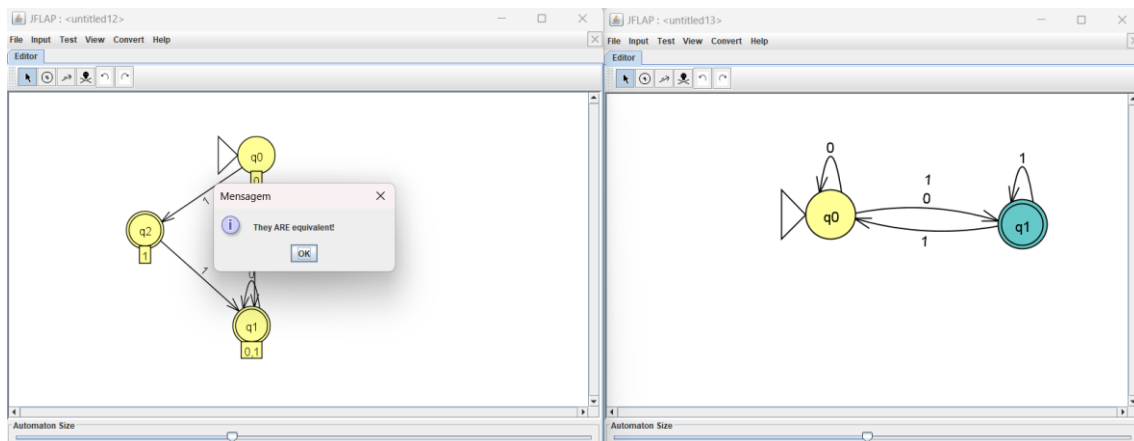
Pegando como exemplo o AFND do exemplo passado e ele após convertemos para AFD, podemos colocar os dois juntos e comparar para ver se são equivalentes. Indo em Finite Automaton e colocando os dois fica assim:



Agora indo em Test e depois em Compare Equivalence ele vai mostrar outros projetos abertos e você seleciona com qual você quer comparar:



Clicando em OK ele nos fala que os dois autômatos são equivalentes:



Slide 16

Exemplo 2.4

Converter uma expressão regular em um autômato finito

Clicando em Grammar nós inserimos a nossa gramática, que é AF $= (\{S, B, QF\}, \{0, 1\}, \delta, S, QF)$, $S \rightarrow 0B$, $B \rightarrow 0$, $B \rightarrow 0B$, $B \rightarrow 1S$:

JFLAP : <untitled1>

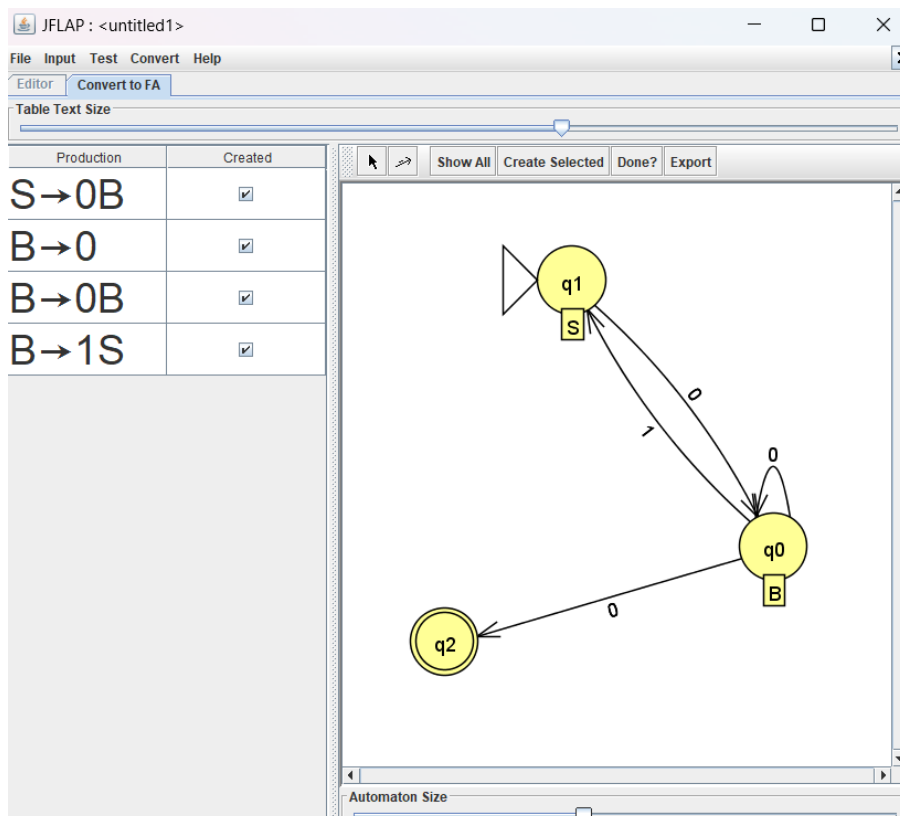
File Input Test Convert Help

Editor

Table Text Size

LHS		RHS
S	→	0B
B	→	0
B	→	0B
B	→	1S

Agora clicando em Convert e depois em Convert Right-Linear Grammar to FA e em Show All ele nos mostra o Autômato Finito resultante:



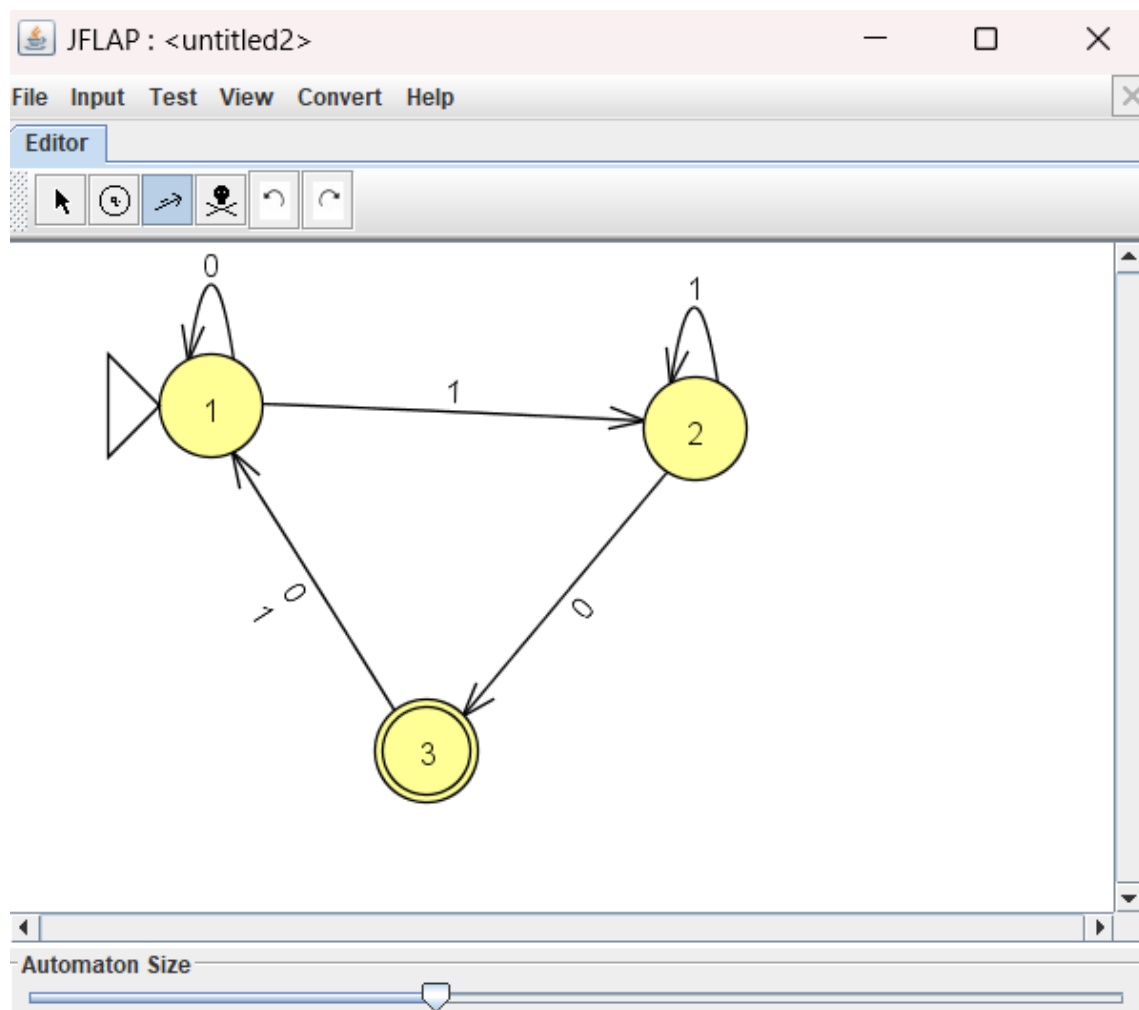
Slide 18

Exemplo Exercício Método Indutivo de E.R

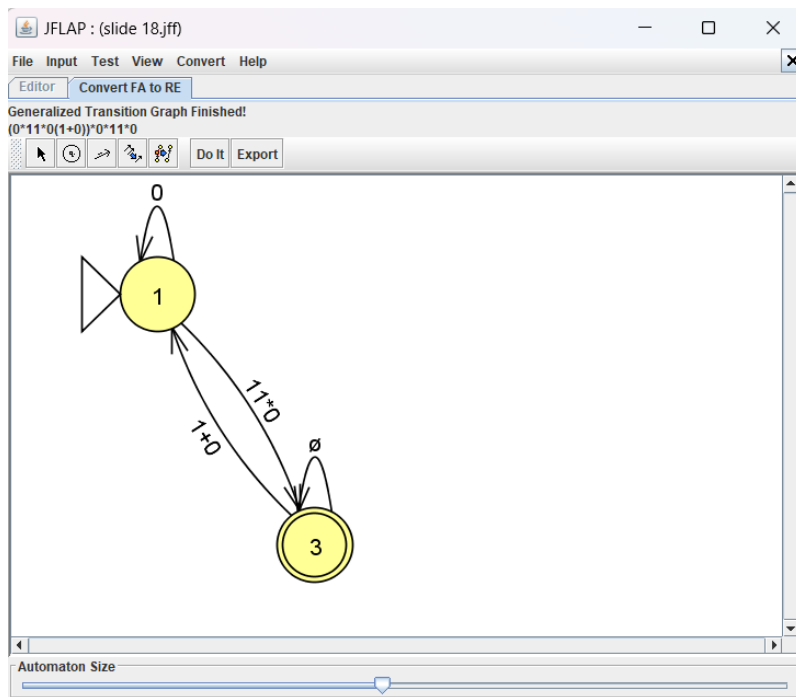
Converter um autômato finito em uma expressão regular

Indo em Finite Automaton e colocando o autômato do exemplo que é $\Sigma = \{0,1\}$, $s_0=1$, $F = \{3\}$ e

$\delta(1,0) = 1$ $\delta(2,1) = 2$ $\delta(1,1) = 2$ $\delta(3,0) = 1$ $\delta(2,0) = 3$ $\delta(3,1) = 1$



Agora clicando em Convert e depois em Convert FA to RE e depois em Do It, o JFLAP converte o autômato para uma expressão regular:

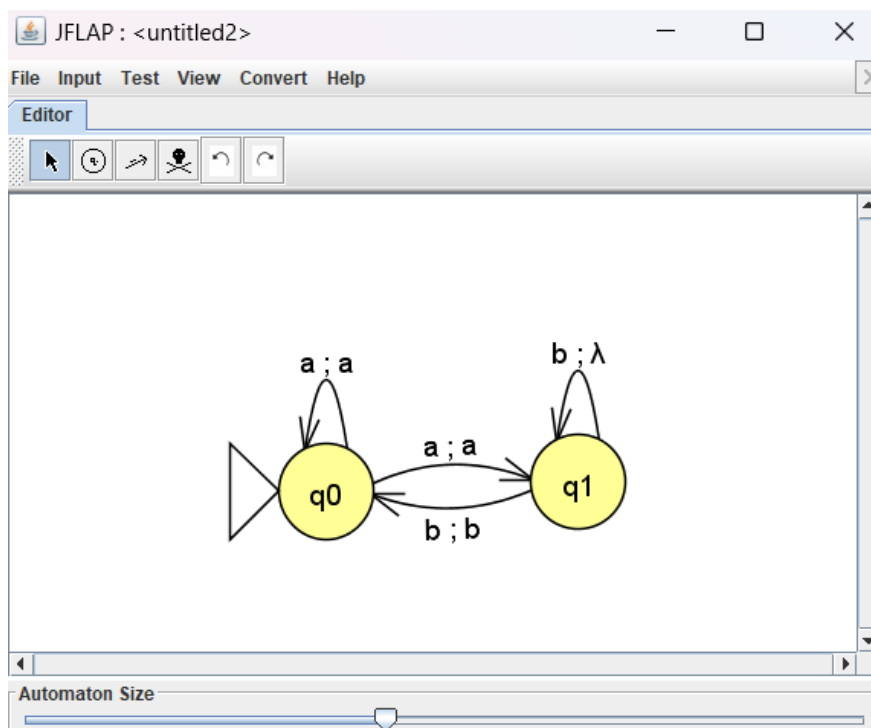


Slide 19

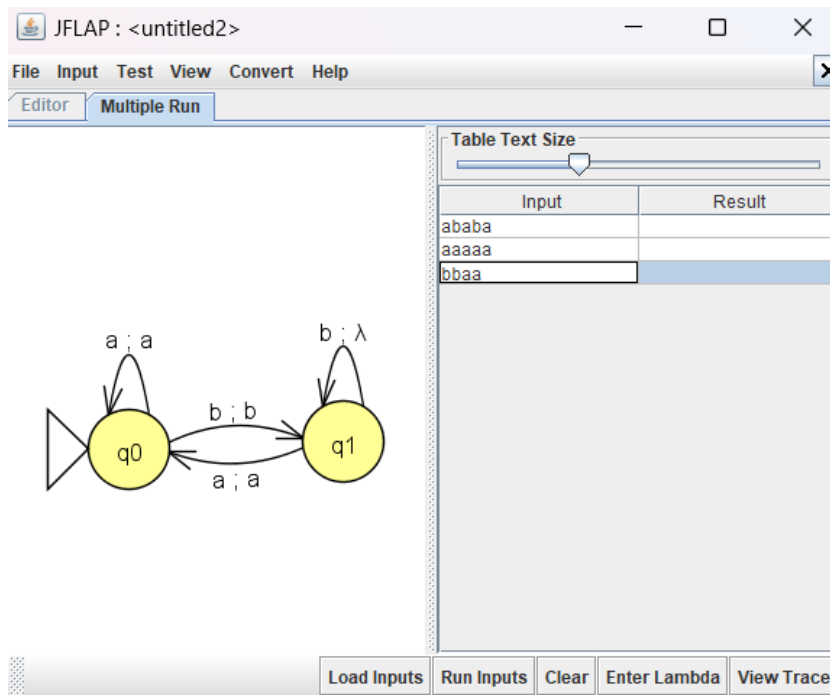
Exemplo 2.8

Criar e testar uma máquina de Mealy

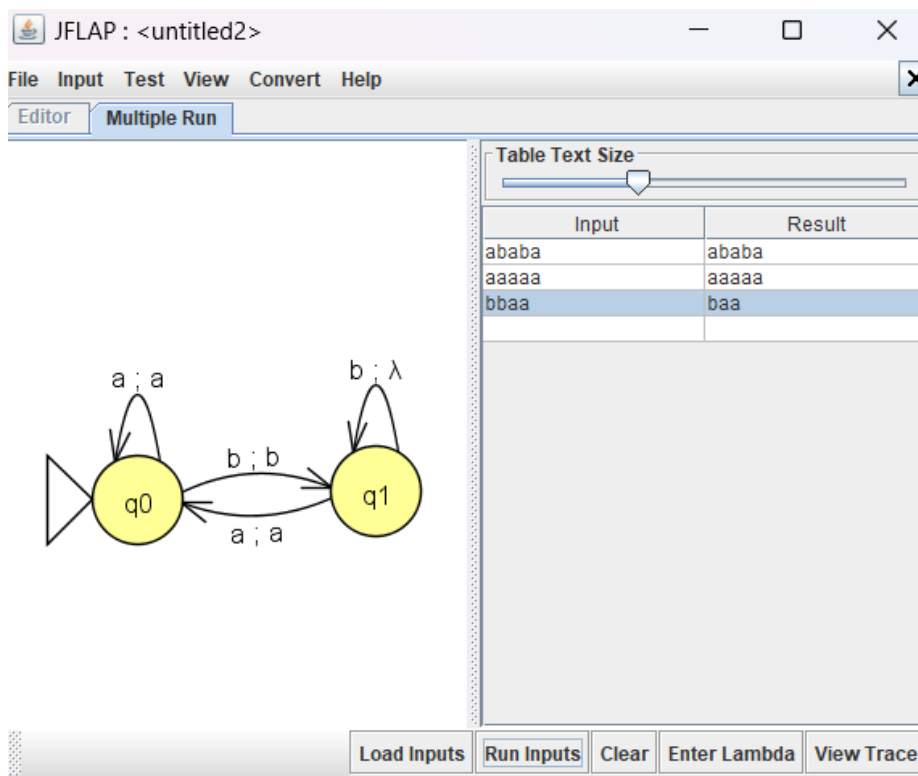
Indo em Mealy Machine, nós podemos criar nossa Máquina de Mealy, colocando à do exemplo 2.8 fica assim:



Para nós testarmos a máquina de Mealy devemos clicar em Input e em Multiple Run para podermos testar vários inputs diferentes, e após inserir os inputs o programa fica assim:



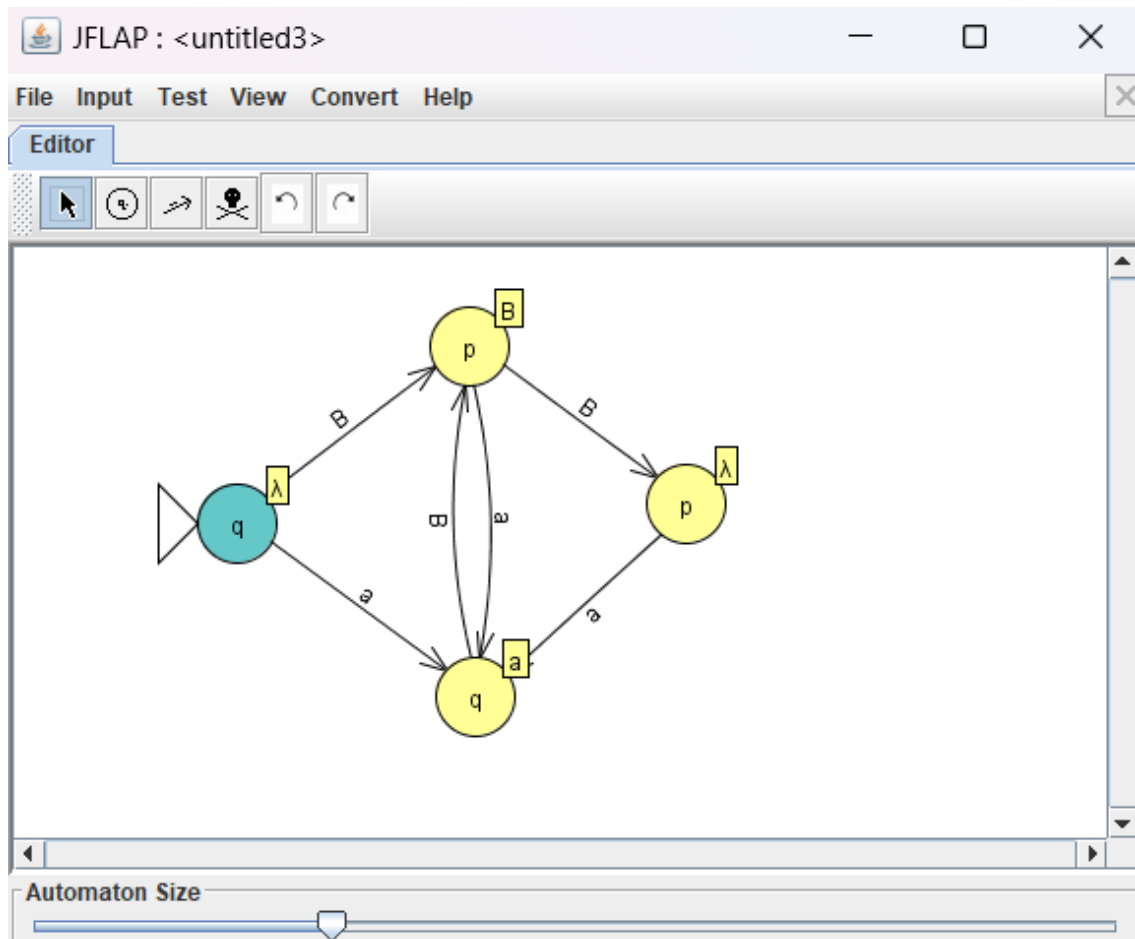
Nesse ponto, com os inputs inseridos, devemos apenas clicar em Run Inputs e todo processo será feito:



Exemplo 2.8

Criar e testar uma máquina de Moore

Indo em Moore Machine, nós podemos criar nossa Máquina de Moore, colocando à do exemplo 2.8 fica assim:



Para nós testarmos a máquina de Moore devemos clicar em Input e em Multiple Run para podermos testar vários inputs diferentes, e após inserir os inputs o programa fica assim:

JFLAP : <untitled3>

File Input Test View Convert Help

Editor Multiple Run

Table Text Size

Input	Result
aBBa	
aaBaBBa	
BBaBa	

Load Inputs Run Inputs Clear Enter Lambda View Trace

Nesse ponto, com os inputs inseridos, devemos apenas clicar em Run Inputs e todo processo será feito:

JFLAP : <untitled3>

File Input Test View Convert Help

Editor Multiple Run

Table Text Size

Input	Result
aBBa	aBa
aaBaBBa	a
BBaBa	BaBa

Load Inputs Run Inputs Clear Enter Lambda View Trace

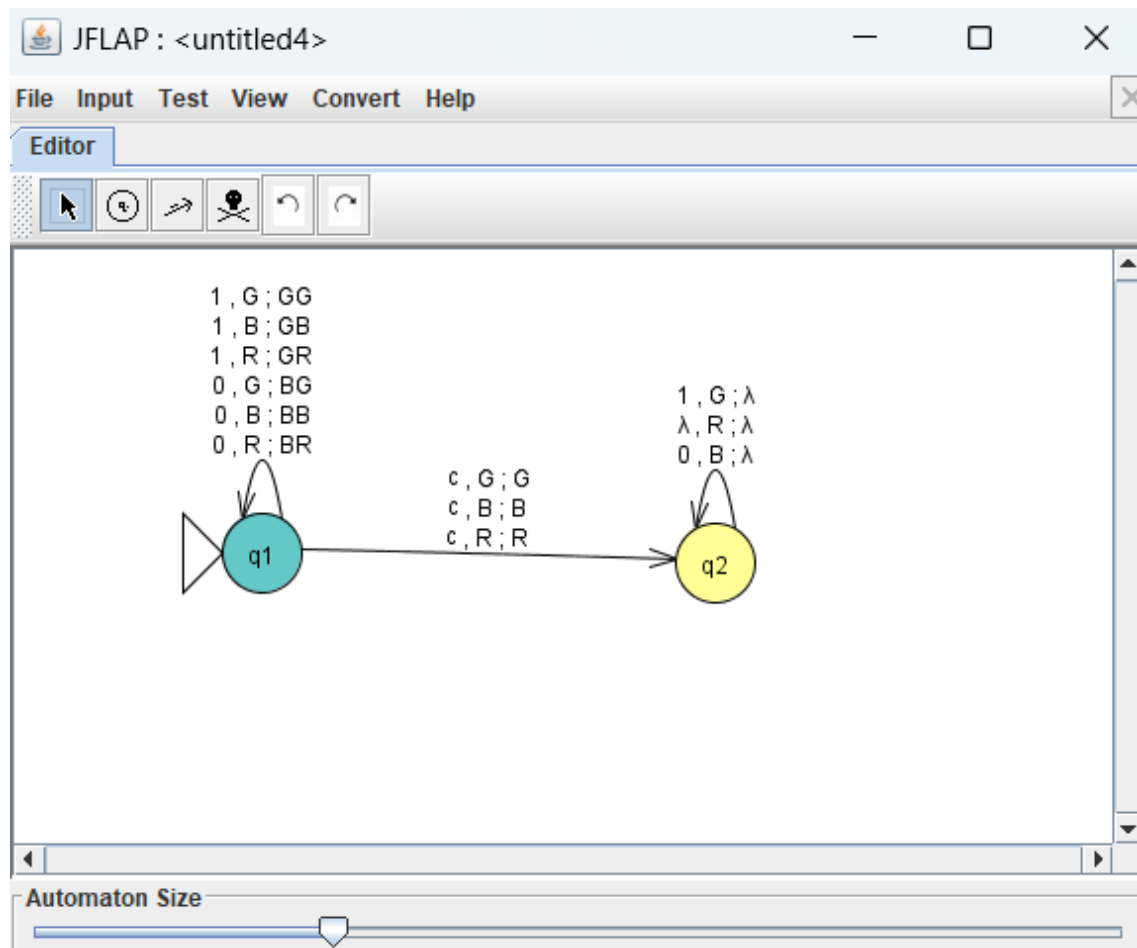
Então ele nos mostra os resultados dos nossos inputs.

Slide 20

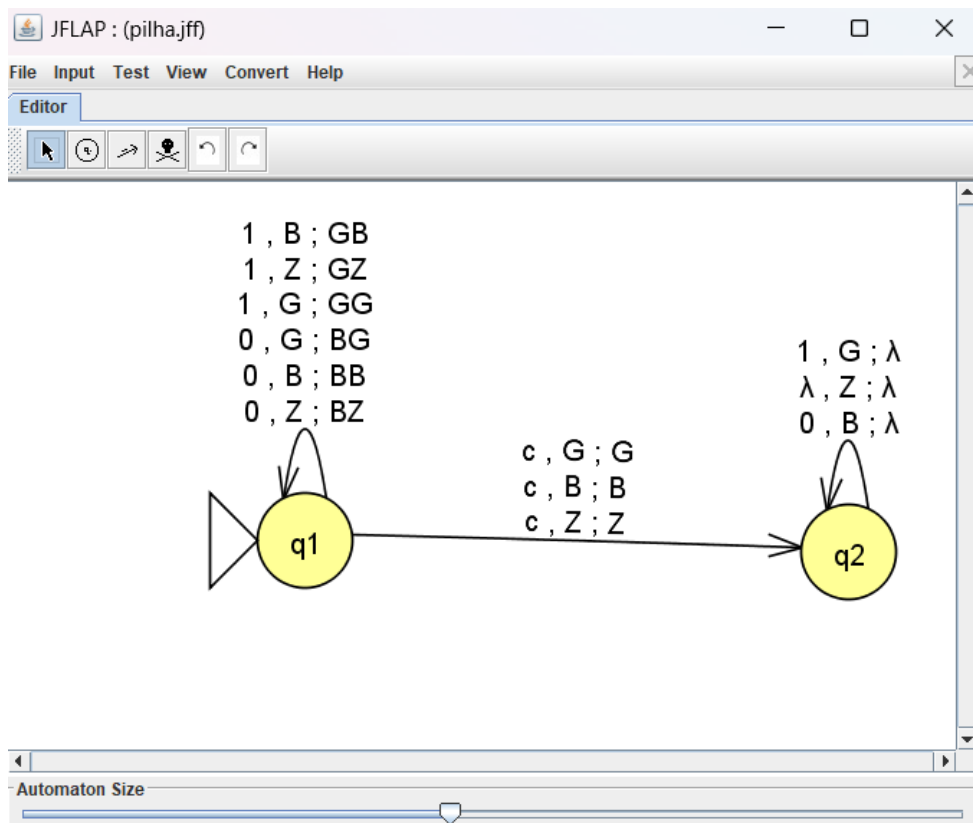
Exemplo 2.9

Criar e testar um autômato de pilha

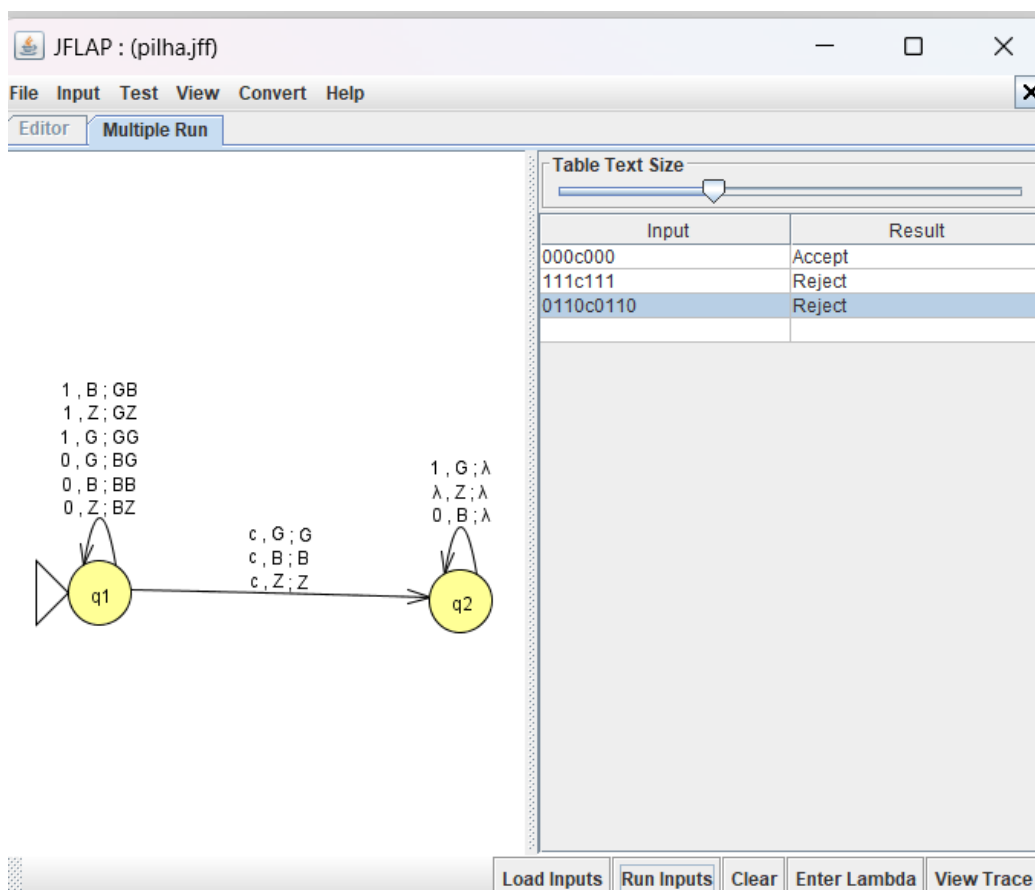
Indo em Pushdown Automaton, nós podemos criar nosso Autômato de Pilha, colocando à do exemplo 2.9 fica assim:



Porém, o JFLAP só reconhece o “Z” como o Símbolo Inicial da Pilha, então tive que “traduzir” o exercício para um jeito que o JFLAP entenda, que é substituindo “R” por “Z”:



Agora podemos clicar em Input e depois em Multiple Run, coloquei alguns inputs como exemplo:



Agora clicamos em Run Inputs e aparece a seguinte tela:

The JFLAP interface shows a finite automaton with two states, q1 and q2. q1 is the start state and q2 is the final state. The transitions are as follows:

- q1 to q1: 1, B; GB; 1, Z; GZ; 1, G; GG; 0, G; BG; 0, B; BB; 0, Z; BZ
- q1 to q2: c, G; G; c, B; B; c, Z; Z
- q2 to q2: 1, G; λ; λ, Z; λ; 0, B; λ

The table of inputs and results is as follows:

Input	Result
000c000	
111c111	
0110c0110	

The 'Input' dialog box is open, showing the 'Accept by' dropdown menu with the following options:

- Final State
- Final State
- Empty Stack

Aqui o JFLAP nós dá a opção de seleccionar se o fim será a pilha chegar em um estado final ou a pilha ficar vazia, eu vou seleccionar a pilha ficar vazia:

The JFLAP interface shows the same finite automaton as before. The table of inputs and results is now updated to show the results for the 'Empty Stack' acceptance criterion:

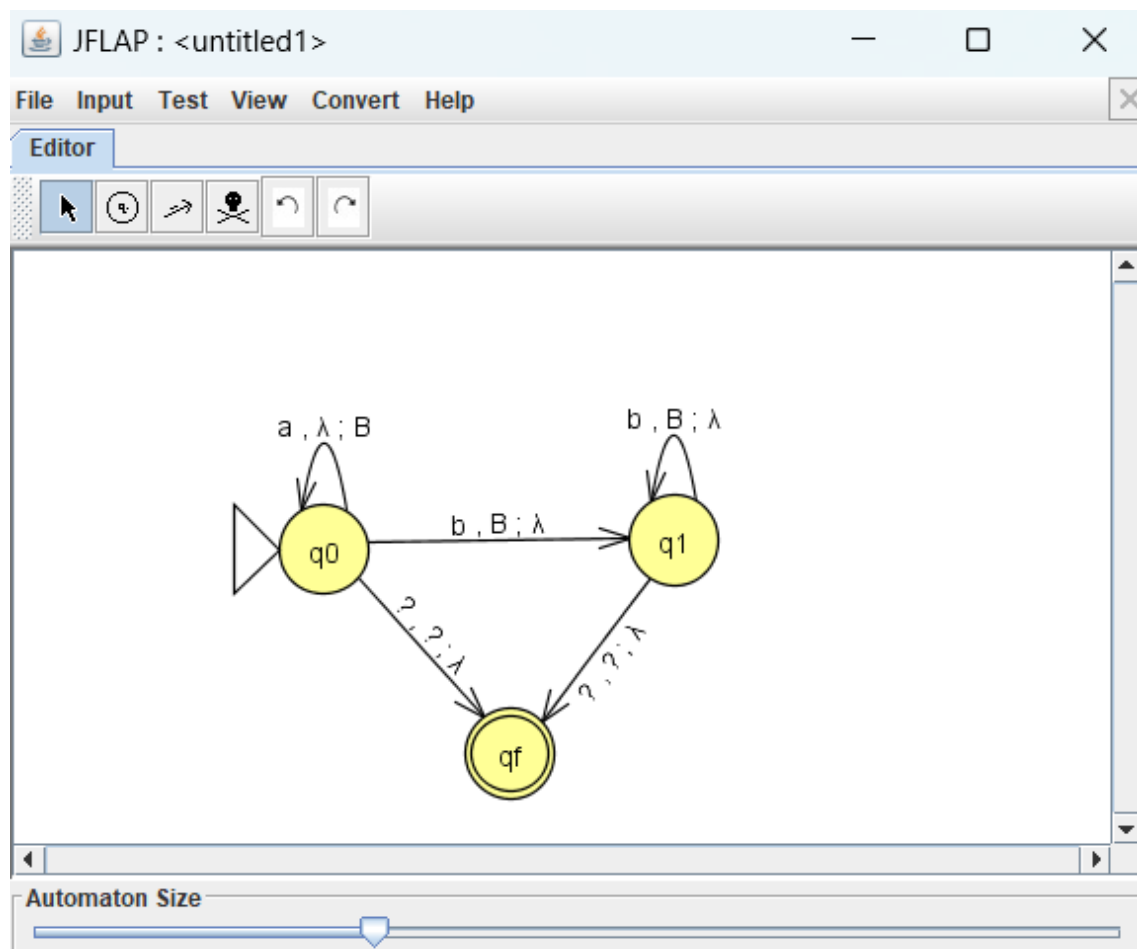
Input	Result
000c000	Accept
111c111	Accept
0110c0110	Accept

Como podemos ver, todas as nossas entradas foram aceitas porque ao final de todas, a pilha fica vazia.

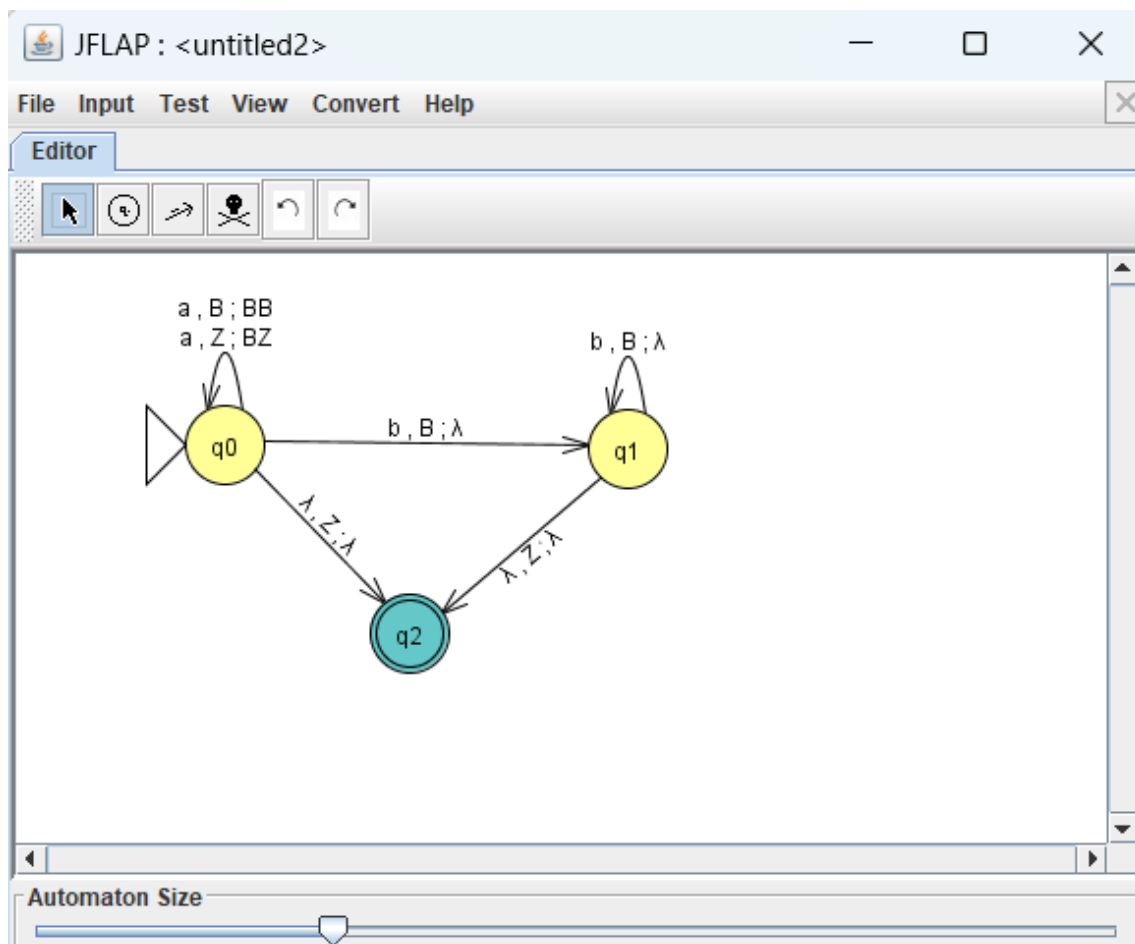
Slide 21

Exemplo 2.10

Indo novamente em Pushdown Automaton, irei colocar o autômato do exemplo no JFLAP:



Porém, o JFLAP não o aceita desse jeito, já que “Z” é o estado inicial padrão do JFLAP e no autômato é “B”, outra coisa é que o JFLAP não aceita “?”, então substituindo tudo ele fica assim:



Agora indo em Input e depois selecionando Multiple Run, podemos clicar em Run Inputs onde eles nós da a seguinte tela:

JFLAP : <untitled2>

File Input Test View Convert Help

Editor Multiple Run

Table Text Size

Input	Result
aabb	

Input

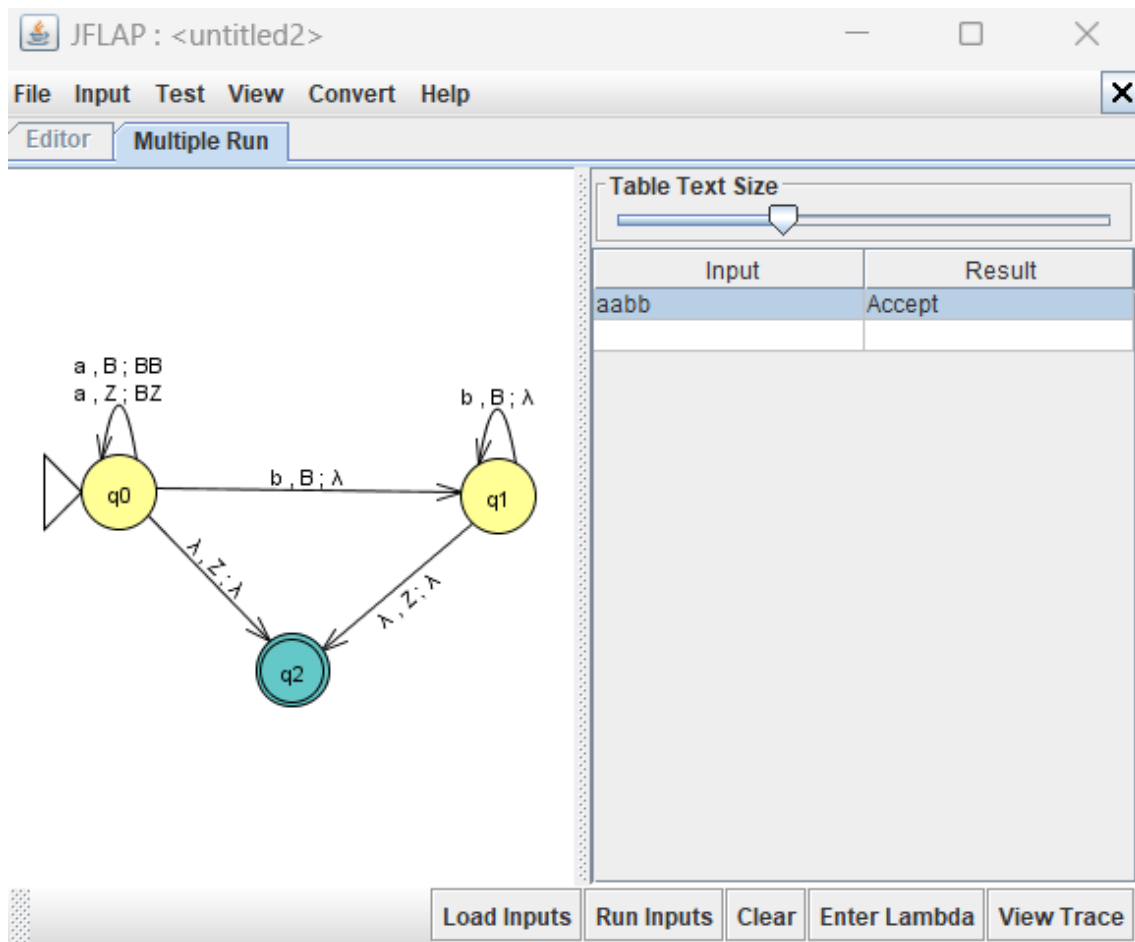
Accept by

Final State

OK Cancelar

Load Inputs Run Inputs Clear Enter Lambda View Trace

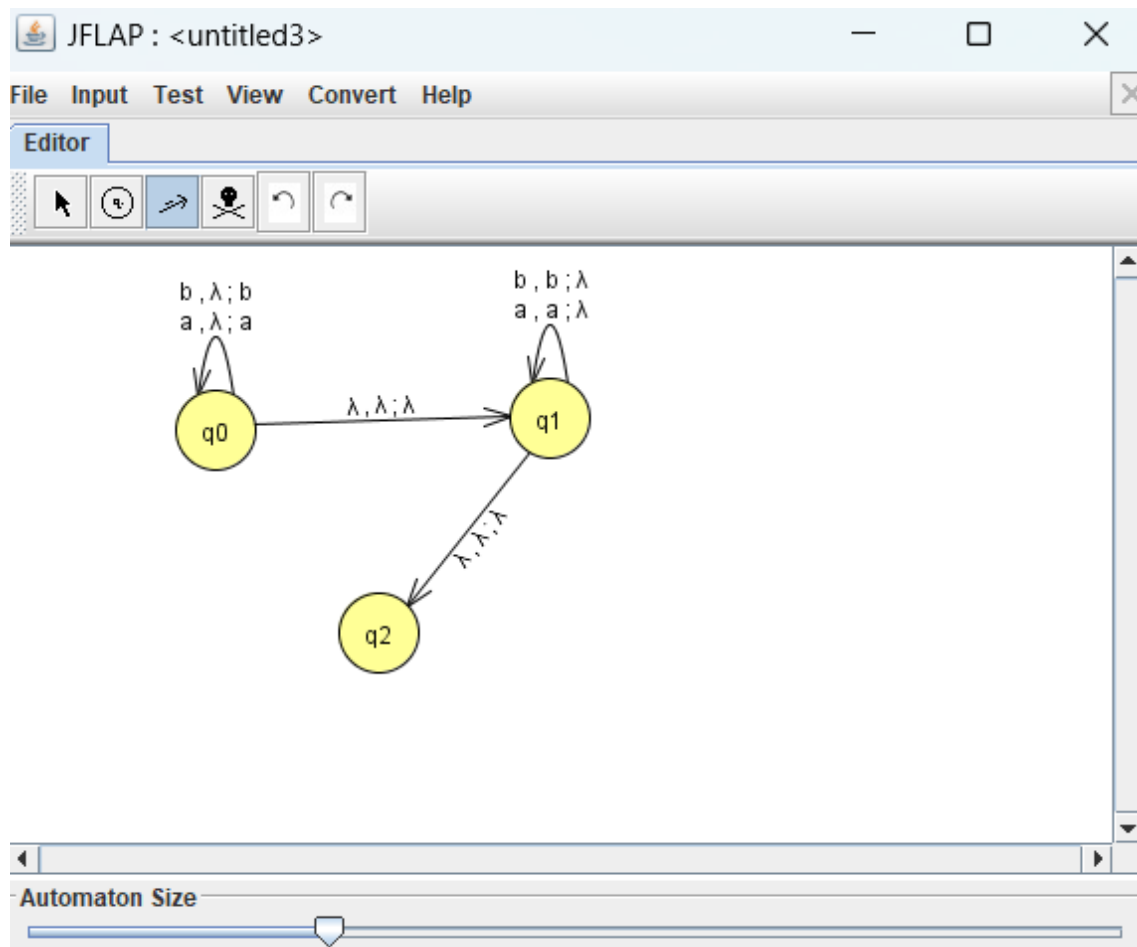
Aqui o JFLAP nos dá a opção de seleccionar se o fim será a pilha chegar em um estado final ou a pilha ficar vazia, eu vou seleccionar a pilha chegar no estado final:



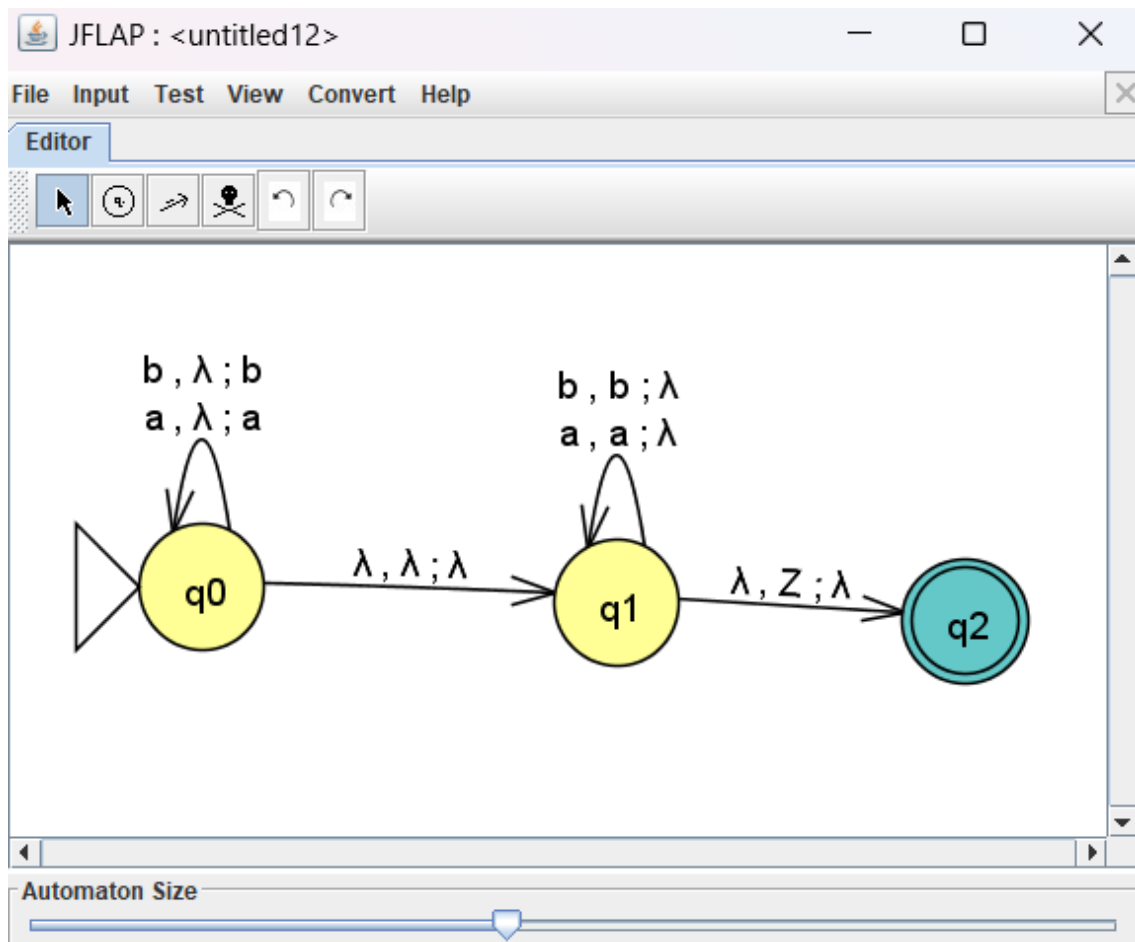
Então eles nos mostra que o input inserido foi aceito.

Exemplo 2.11

Indo novamente em Pushdown Automaton, irei colocar o autômato do exemplo no JFLAP:



Novamente, o JFLAP não o aceita desse jeito, já que “Z” é o estado inicial padrão do JFLAP, outra coisa é que o JFLAP não aceita “?”, então substituindo tudo ele fica assim:



Agora indo em Input e depois selecionando Multiple Run, podemos clicar em Run Inputs onde eles nós da a seguinte tela:

JFLAP : <untitled12>

File Input Test View Convert Help

Editor Multiple Run

Table Text Size

Input	Result
aaaa	

Input

Accept by

Final State

OK Cancelar

Load Inputs Run Inputs Clear Enter Lambda View Trace

```

graph LR
    start(( )) --> q0((q0))
    q0 -- "b, λ; b  
a, λ; a" --> q0
    q0 -- "λ, λ; λ" --> q1((q1))
    q1 -- "b, b; λ  
a, a; λ" --> q1
    q1 -- "λ, Z; λ" --> q2(((q2)))
  
```

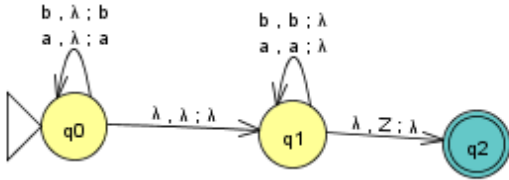
Aqui o JFLAP nos dá a opção de seleccionar se o fim será a pilha chegar em um estado final ou a pilha ficar vazia, eu vou seleccionar a pilha chegar no estado final:

JFLAP : <untitled12>

File Input Test View Convert Help

Editor Multiple Run

Table Text Size



Input	Result
aaaa	Accept

Load Inputs Run Inputs Clear Enter Lambda View Trace

Então ele nos mostra que o input inserido foi aceito.

Comparando o JFLAP com os slides da professora Cinthyan, pude perceber que a maioria das coisas que ela nos ensinou, contém no JFLAP, porém, na minha percepção, os slides explicam melhor cada coisa do que o JFLAP. O JFLAP em si não nos ensina nada, você tem que entrar no site de tutorial para poder ler o que cada coisa faz no sistema dele, mas todo o tutorial está em inglês, demandando um conhecimento à mais do que o necessário para compreender os slides. Ele serve muito bem para testar exemplos ou criar os seus próprios para estudar, porém, na questão de ensino, os slides são melhores.