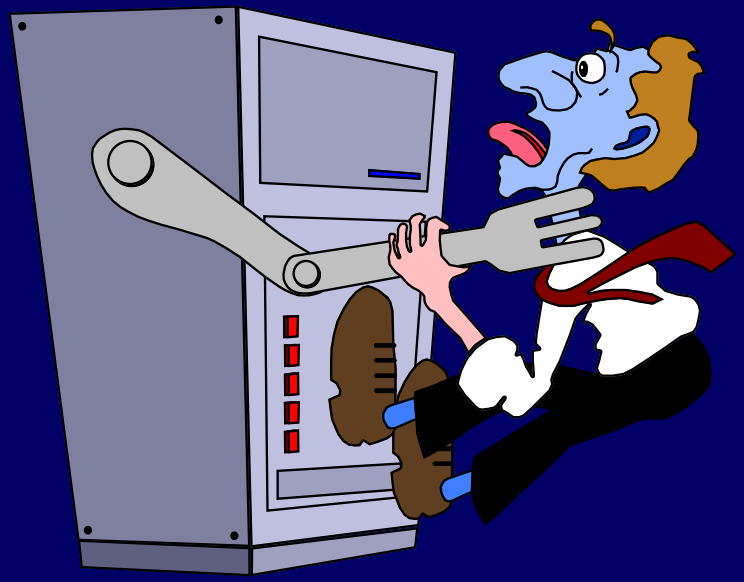
Máquinas de Turing 3



USOS de uma Máquina de Turing

Usos de uma MT

- como reconhecedor de linguagens (Visto)
- para calcular funções
- para processar problemas de decisão (procedimento)

MT como um processador de funções inteiras

- Tradicionalmente, os inteiros são representados em unário
- O inteiro i >= 0 é representado pela cadeia 0^i.
- Se a função tem k argumentos (i1, i2, ..., ik) então esses inteiros são colocados na fita separados por 1´s como:

$$0^{i1}\ 1\ 0^{i2}\ 1\ \ldots\ 1\ 0^{ik}$$

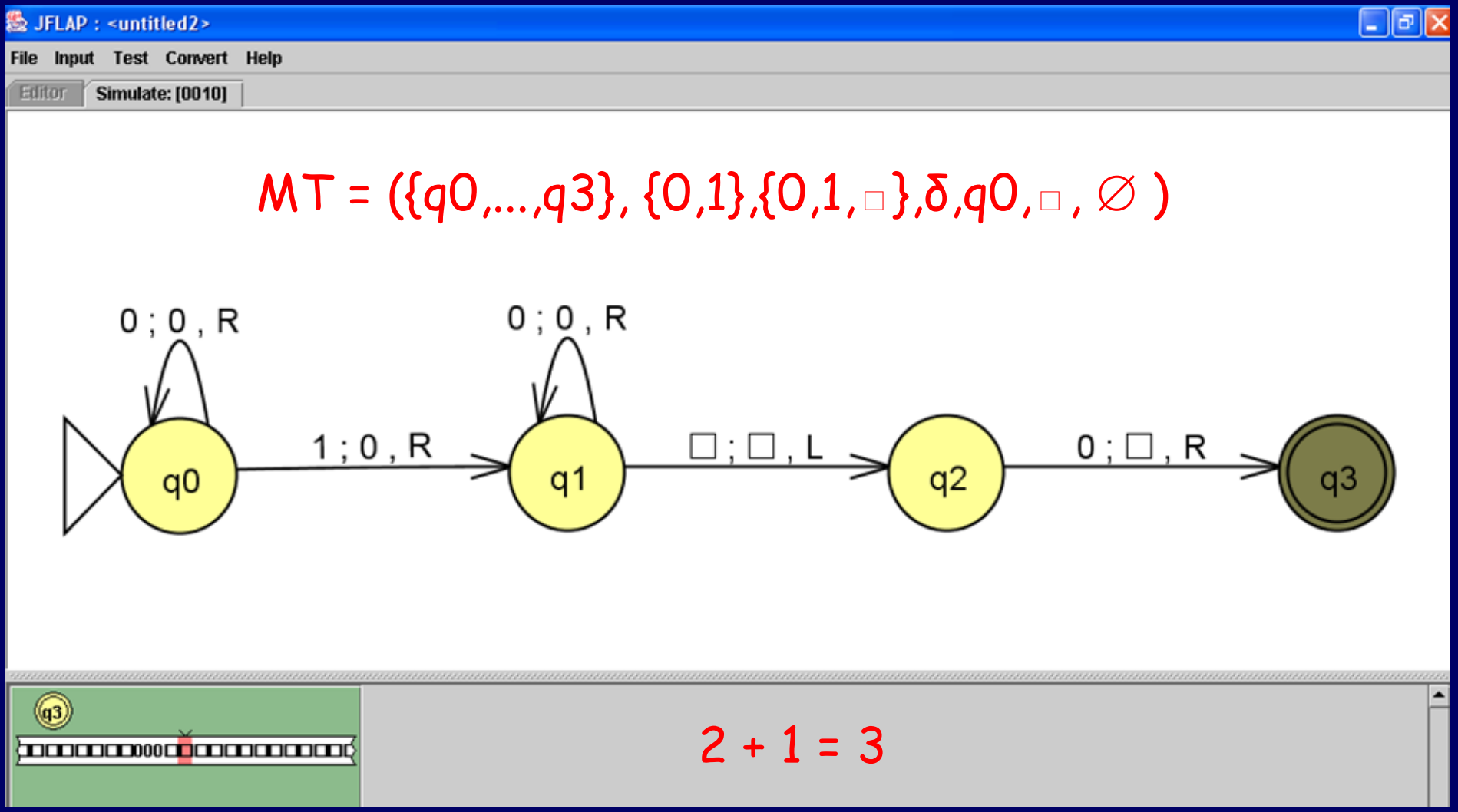
- O inverso também é possível.

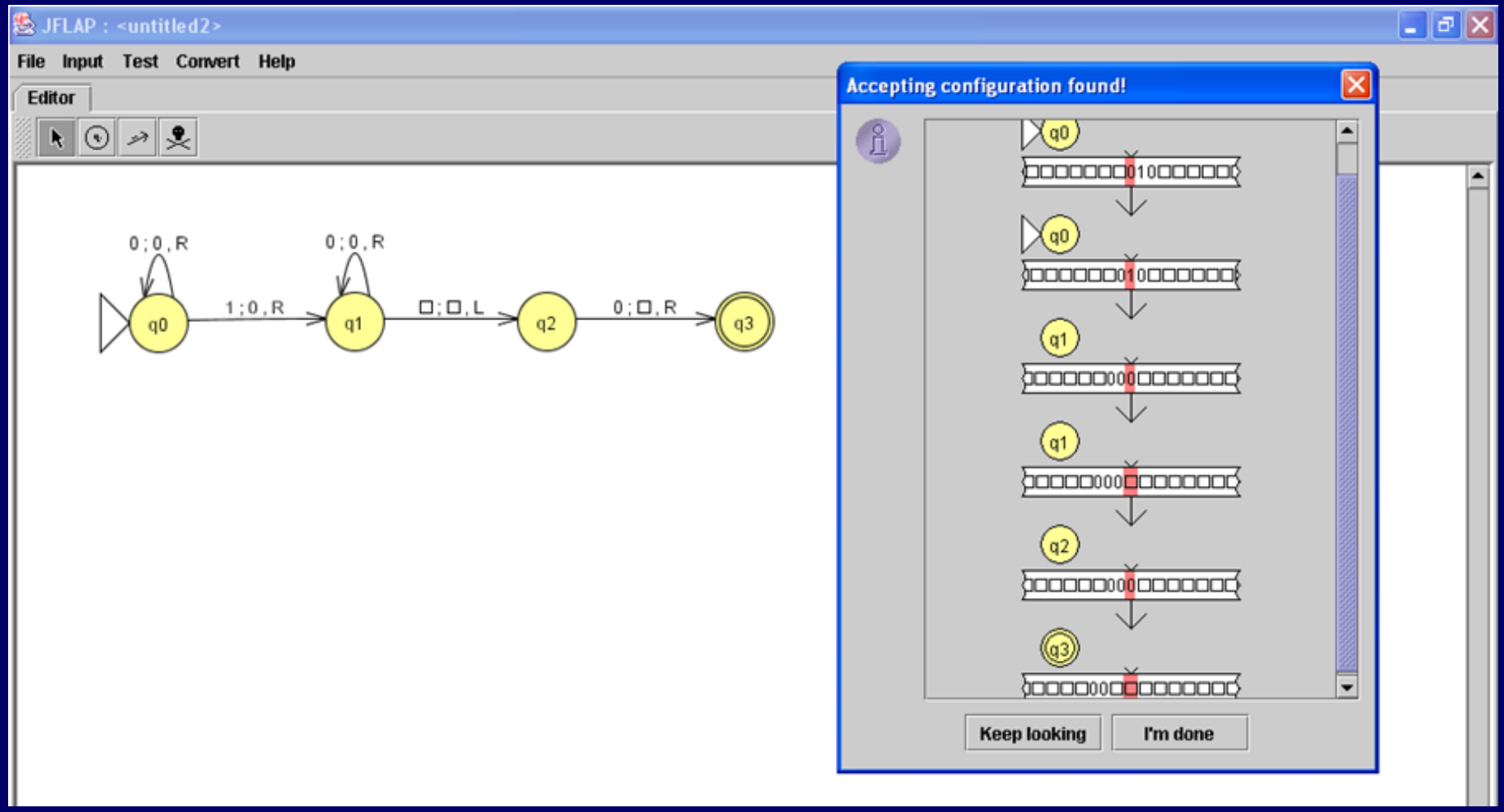
- Se a máquina pára (não importa se num estado final) com a fita consistindo de 0^m para algum m então dizemos que f(i1,i2,...ik) = m, onde f é uma função de k argumentos computados por essa MT.

1. MT que soma dois números naturais não negativos

- 1. Codifique a entrada como $0^a\ 1\ 0^b$
- 2. A MT deve parar com 0^{a+b} na fita

- Casos de teste:
- 0 + 0
- 1 + 3
- 0 + 4

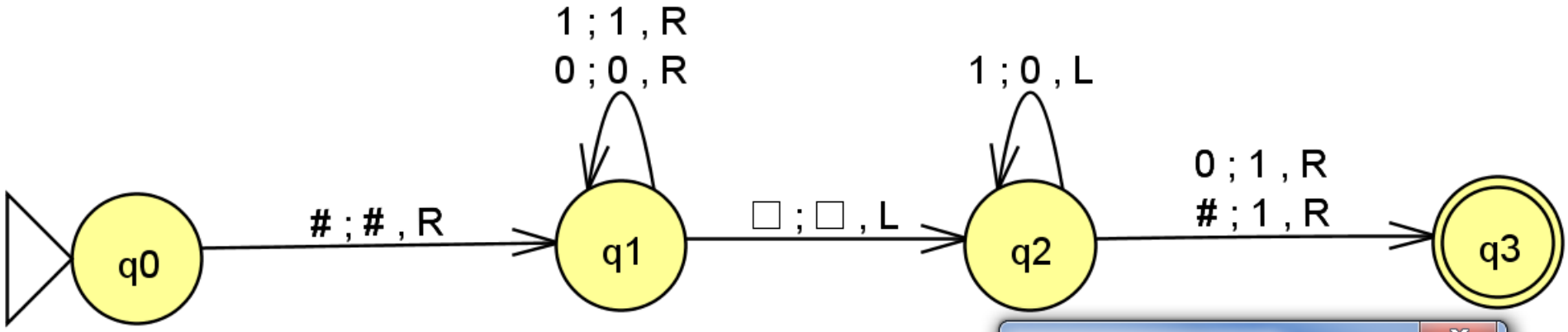

JFLAP : <untitled2>
File Input Test Convert Help
Editor Simulate: [0010]
MT = ({q0,...,q3}, {0,1},{0,1,□},δ,q0,□ , ∅)
0 ; 0 , R
0 ; 0 , R
1 ; 0 , R
□ ; □ , L
0 ; □ , R
q0
q1
q2
q3
q3
2 + 1 = 3


JFLAP : <untitled2>
File Input Test Convert Help
Editor
0 ; 0 , R
0 ; 0 , R
1 ; 0 , R
□ ; □ , L
0 ; □ , R
q0
q1
q2
q3
Accepting configuration found!
Keep looking
I'm done

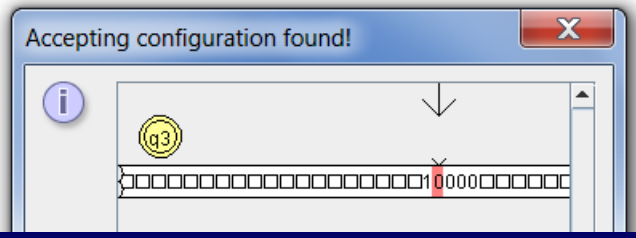
- 2. MT que faz adição de 1 unidade (soma 1) em números representados em binário, começados por #

- Considerem que os números podem ter zeros a esquerda: #00011, por exemplo.

- Casos de Teste:
- #100 → #101
- #0010 → #0011
- #111 → 1000
- #1111 → 10000



$MT = (\{q0,\ldots,q3\}, \{\#,0,1\},\{\#,0,1,□\},\delta,q0,□,\varnothing)$

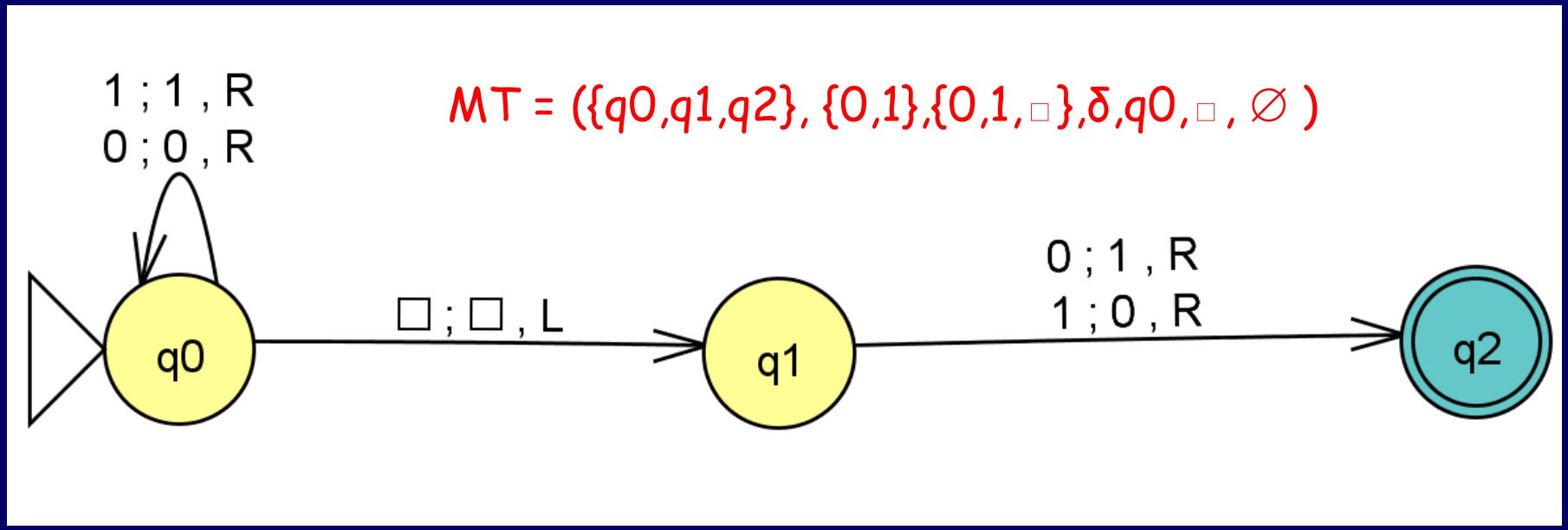


3. Faça uma Máquina de Turing que aceita números binários e faça:

a. Se o número for impar, subtrai 1

b. Se o número for par, adiciona 1

- Casos de Teste:
- 101 → 100
- 1010 → 1011


1 ; 1 , R
0 ; 0 , R
MT = ({q0,q1,q2}, {0,1},{0,1, □},δ,q0, □ , ∅)
□ ; □ , L
0 ; 1 , R
1 ; 0 , R
q0
q1
q2

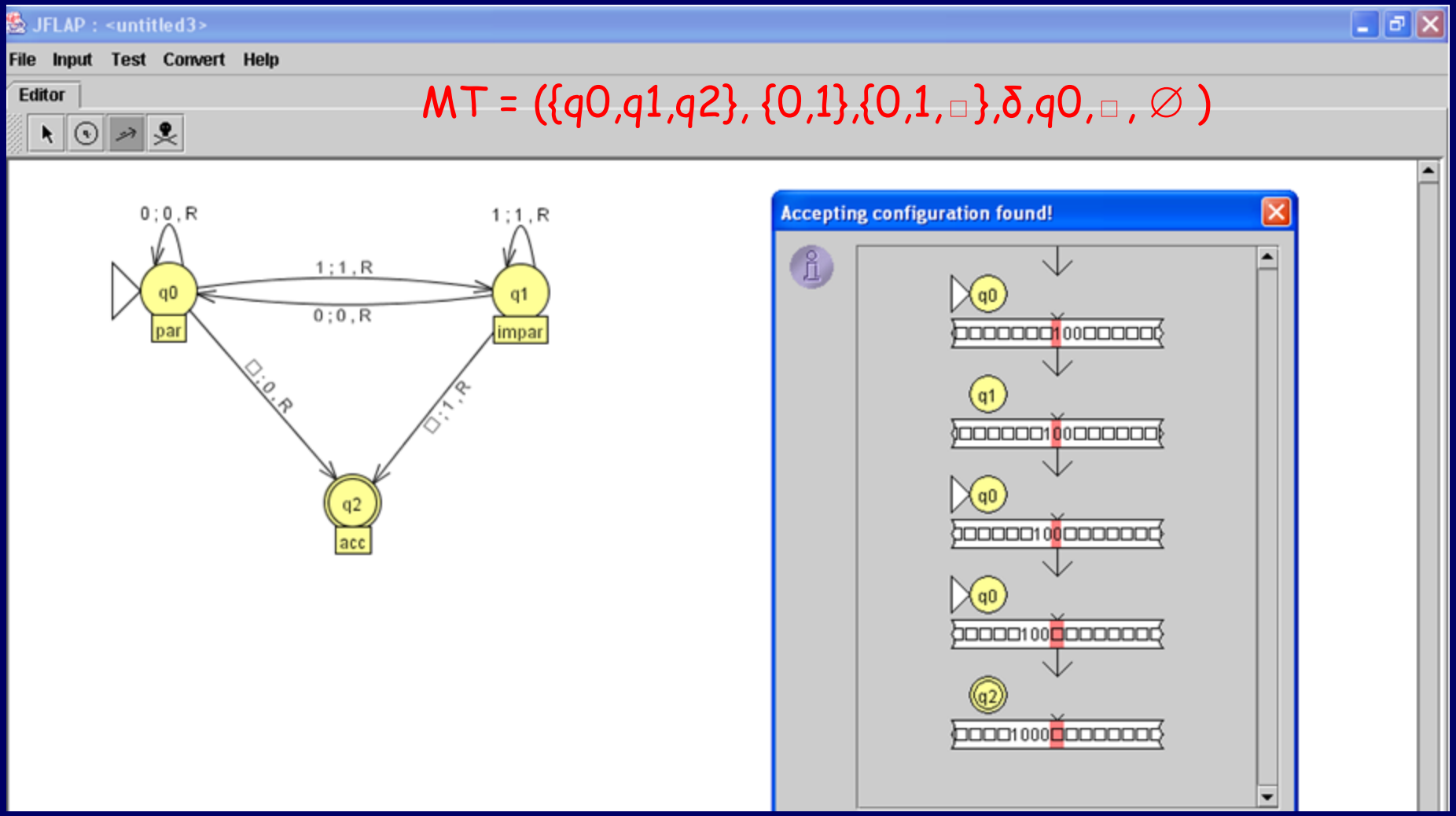
MT para processar problemas de decisão

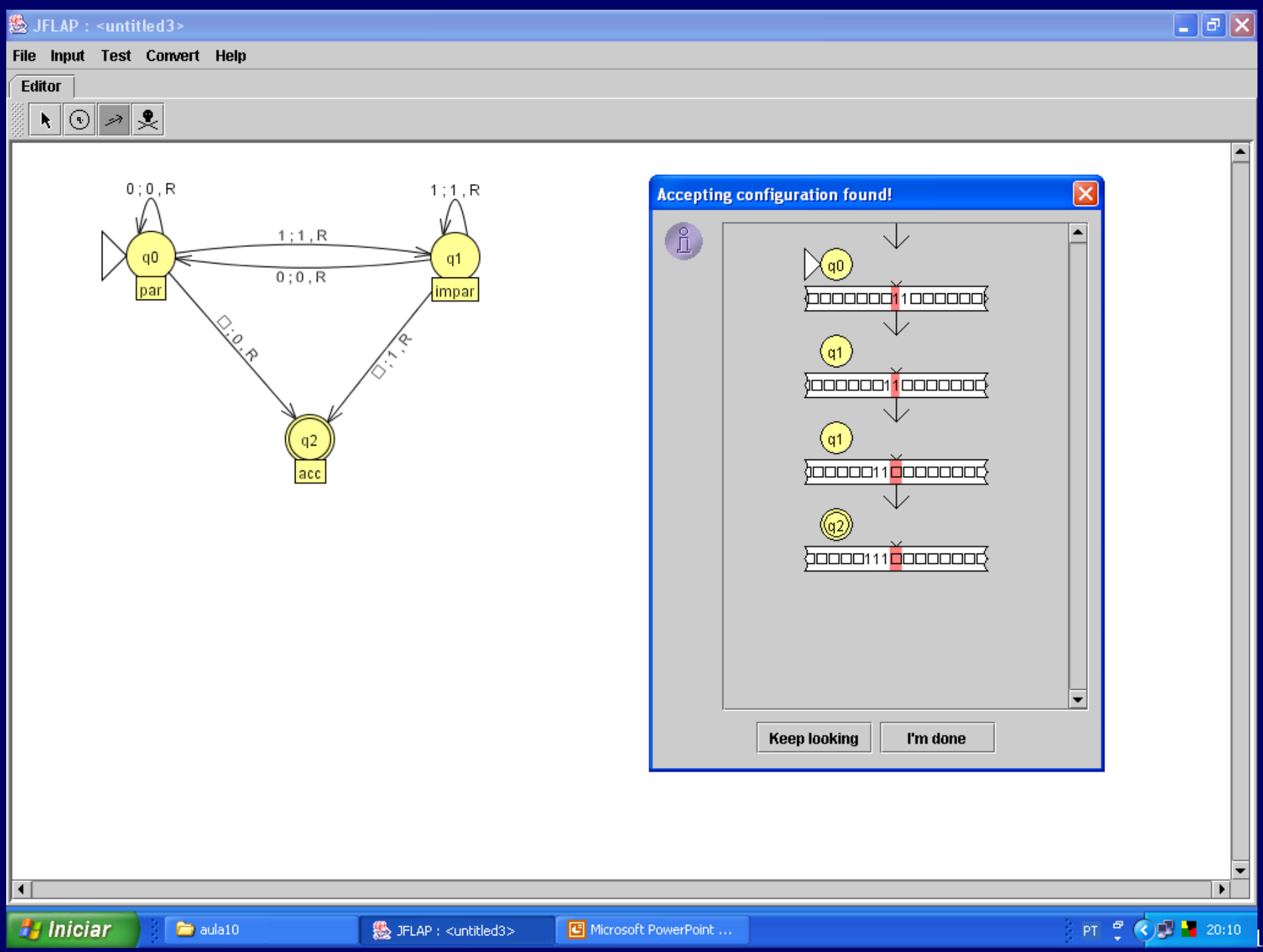
- Quando usamos a MT para decidir (responder T/F) alguma propriedade, depois que a máquina parar, olhamos na fita a procura do resultado 0/1 após a cadeia de entrada: algoritmo.

1. MT que decide se um número binário é par ou impar

- Fica em q0 enquanto par
- Fica em q1 enquanto impar

- Se Par - pára e escreve 0
- Se Impar - pára e escreve 1


JFLAP : <untitled3>
File Input Test Convert Help
Editor
MT = ({q0,q1,q2}, {0,1},{0,1,□},δ,q0,□, ∅)
0 ; 0 , R
1 ; 1 , R
1 ; 1 , R
q0
par
1 ; 1 , R
0 ; 0 , R
q1
impar
□ ; 0 , R
□ ; 1 , R
q2
acc
Accepting configuration found!
q0
100
q1
100
q0
100
q0
100
q2
1000


JFLAP : <untitled3>
File Input Test Convert Help
Editor
0 ; 0 , R
1 ; 1 , R
q0
par
1 ; 1 , R
0 ; 0 , R
q1
impar
□ ; 0 , R
□ ; 1 , R
q2
acc
Accepting configuration found!
q0
11
q1
11
q1
11
q2
111
Keep looking
I'm done
Iniciar
aula10
JFLAP : <untitled3>
Microsoft PowerPoint ...
PT
20:10

Exercícios

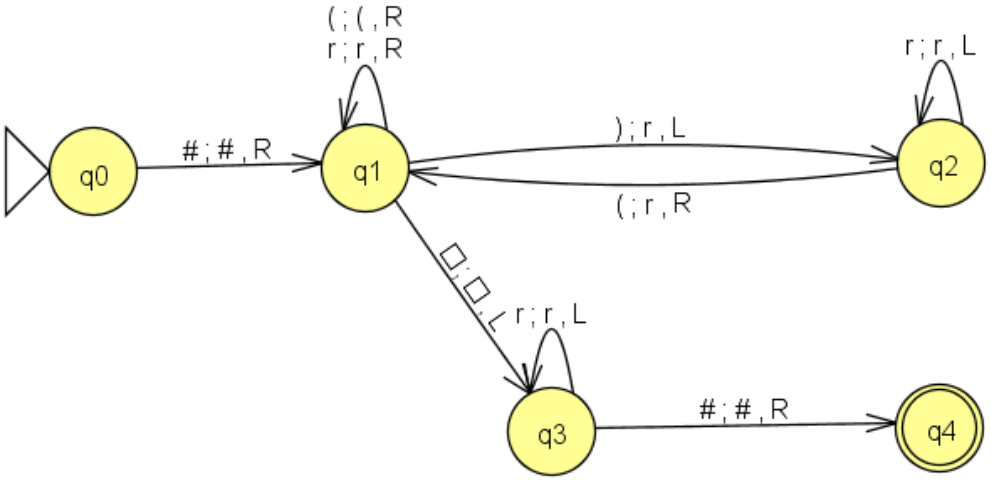
1) Construir uma MT que decida se uma seqüência de parênteses é bem formada.
   - Escreve 0 se mal formada
   - Escreve 1 se bem formada

- Dica: considere que a cadeia de parênteses é iniciada por # para facilitar a checagem.

- Idéia: Procurem por um) e substitua por X e em seguida voltar a esquerda procurando o (mais próximo para substituir por X também.

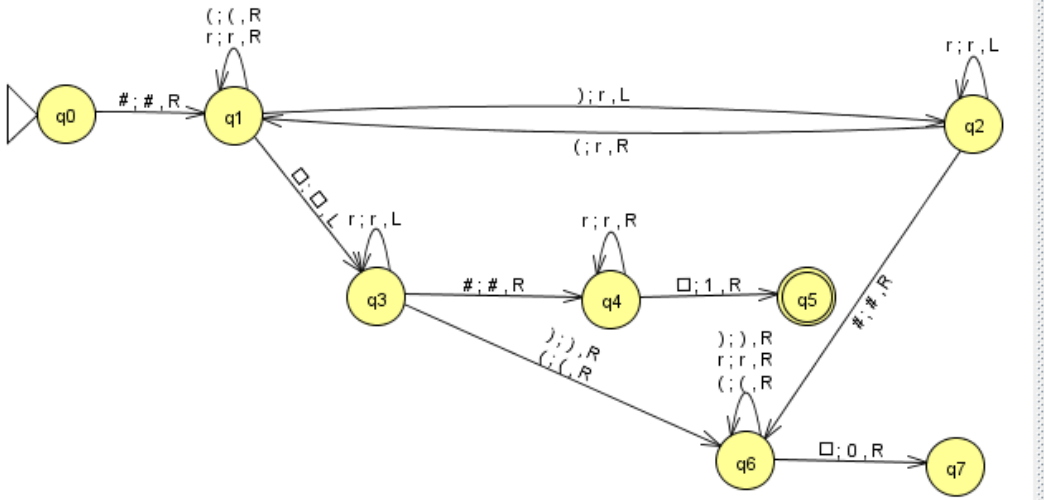
2) Construir uma MT que dada uma cadeia w pertencente ao fecho de {0,1} duplique w. Quando a máquina parar a fita deve conter w#w sendo que # indica fim de w.

Funciona como reconhecedor

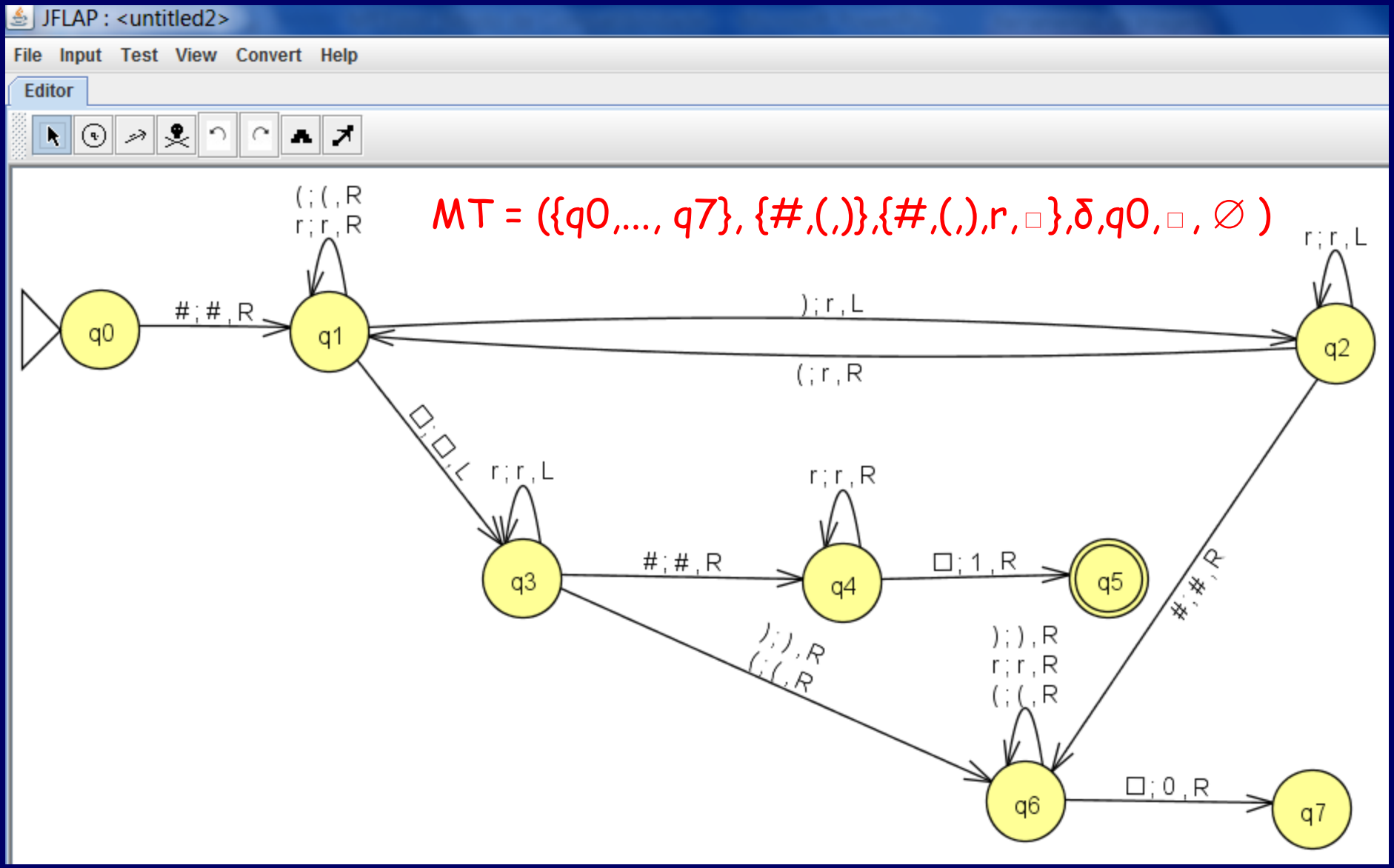


Input	Result
#()	Accept
#	Accept
#(()	Reject
#(())	Accept
#(((	Reject
#()()()	Accept
#)))	Reject
#())	Reject
#()(	Reject

Funciona como algoritmo



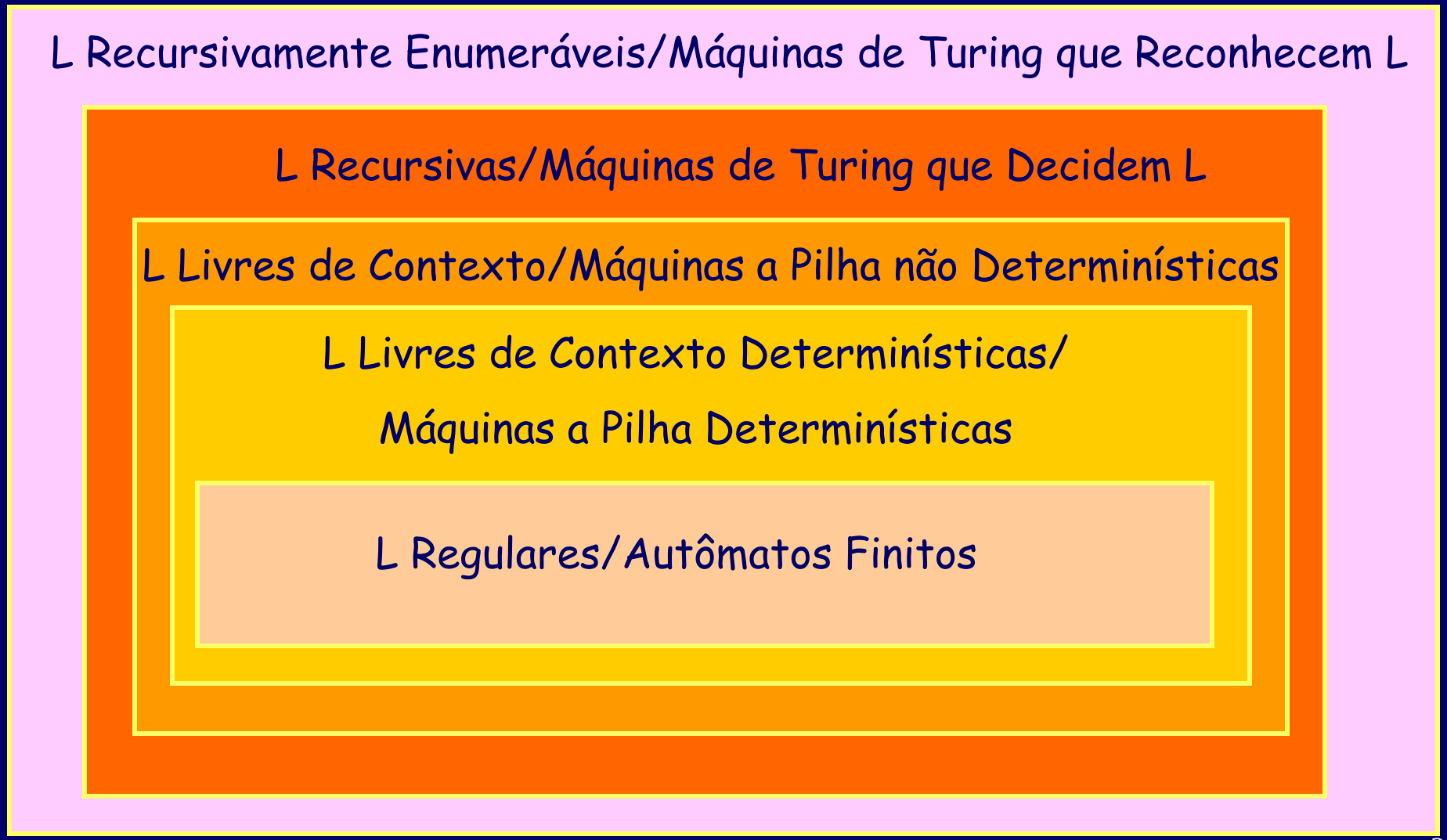
Input	Result
#()	Accept
#	Accept
#(()	Reject
#(())	Accept
#(((	Reject
#()()()	Accept
#)))	Reject
#())	Reject
#)(	Reject


JFLAP : <untitled2>
File Input Test View Convert Help
Editor
MT = ({q0,..., q7}, {#,(,)},{#,(,),r,□},δ,q0,□, ∅)
q0
q1
q2
q3
q4
q5
q6
q7
#;#,R
(;(,R
r;r,R
);r,L
(;r,R
r;r,L
□;□,L
r;r,R
#;#,R
□;1,R
);),R
(;(,R
);),R
r;r,R
(;(,R
#;#,R
□;0,R

Lembrem que:

- MT para calcular funções:
  - Se f(n)
    - não é definida para todo n → f(n) é função parcial;
    - se sempre é definida é função total
  - Se MT computa f(n)
    - para todo n → f(n) é recursiva (algoritmo que computa f(n))
    - sempre que f(n) é definida mas NÃO pára para aqueles n para os quais f(n) não é definida → f(n) é parcialmente recursiva (ou recursivamente enumerável)
- MT para processar problemas de decisão:
  - Quando consideramos a MT como procedimento, se ela pára para toda a entrada dizemos que o procedimento é um algoritmo.

Hierarquia das Classes de Máquinas e Linguagens



Onde estão as LSC na figura anterior???

- As LSC são reconhecidas por MT com fita limitada, mas na Teoria da Computabilidade o foco está nas MT que decidem e nas MT que reconhecem.

- Esta é a razão de serem menos comentadas.
- As LSC são um subconjunto próprio das L Recursivas.

Máquinas e Linguagens incluindo as LSC