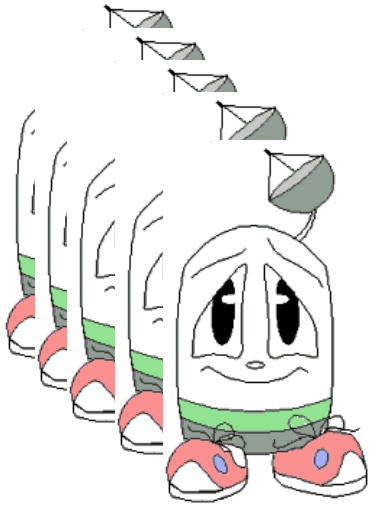
Autômatos Finitos



AF Não-determinísticos
Equivalência entre AFND e AFD

AF NÃO-Determinístico (AFND)

- Consideremos uma modificação no modelo do AFD para permitir
  - zero, uma ou mais transições de um estado sobre o MESMO símbolo de entrada.
  - Ou visto de outra forma:
  - a função toma um estado e uma entrada e devolve zero, um ou mais estados.
- Esse modelo é chamado AFND.

Por que isso é interessante?

- Pois possibilita tentar alternativas distintas
- Se cada estado representa uma opção, ir para mais de um estado representa tentar opções diferentes de caminho para a solução

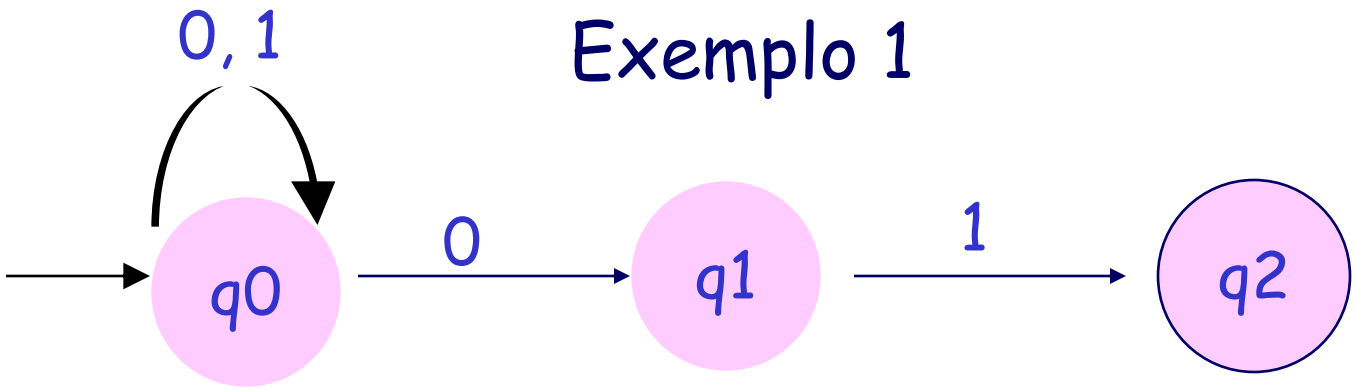
AFND

- Uma cadeia de entrada $a_1a_2...a_n$ é aceita/reconhecida por um AFND
  - se existe AO MENOS UMA sequência de transições que leva do estado inicial para algum estado final.

- Ele funciona como se houvesse a multiplicação da unidade de controle, uma para cada alternativa,
  - processando independentemente, sem compartilhar recursos com as demais,
  - aceitando a cadeia se ao menos uma delas parar num estado final.

- Pensem: Será que o não-determinismo aumenta o poder de reconhecimento de linguagens de um AF?

Exemplo 1



Esse autômato aceita cadeias de 0's e 1's que terminam em 01. Analisem a aceitação de "00101"

Quando está em qo e o símbolo lido é 0 ele tem a opção de:

continuar em qo no caso do fim da cadeia não estar próximo

OU

ir para q1 porque aposta que o fim está chegando.

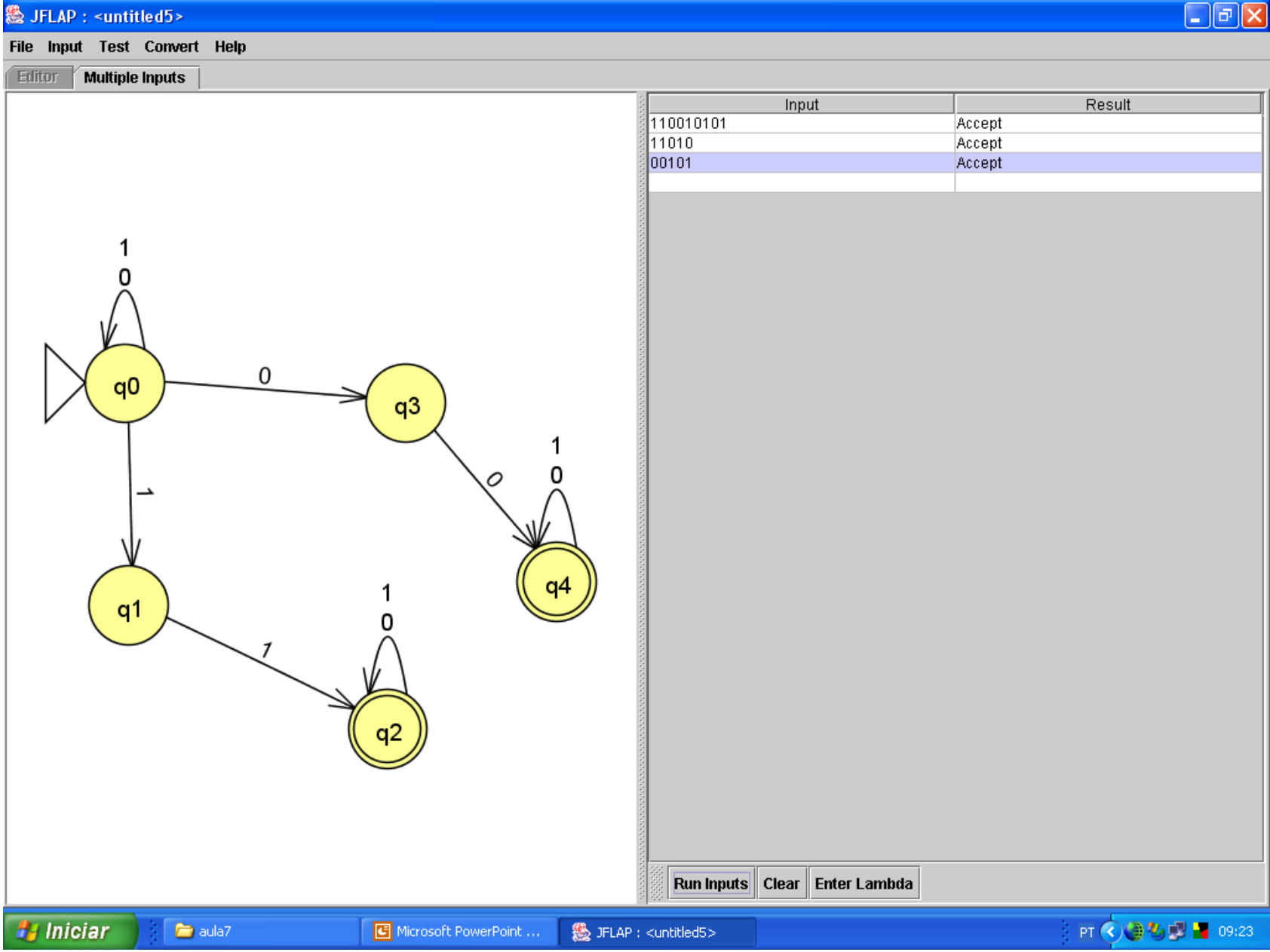
E na verdade ele executa as duas opções!

Por isso costumamos pensar que ele "advinha" qual é a alternativa correta entre muitas.

Exemplo 2

- L(M) = {x $\in${0,1}* | x tenha dois 0's consecutivos OU dois 1's consecutivos}

Ex 2



$L(M) = \{x \in \{0,1\}^* \mid x \text{ tenha dois 0's consecutivos OU dois 1's consecutivos}\}$

$M = (\{q0,q1,q2,q3,q4\},\{0,1\},\delta,q0,\{q2,q4\})$

Definição Formal de um AFND

Denotamos um AFND pela 5-tupla (Q,Σ,δ, q_0,F) onde Q,Σ, q_0 e F são os mesmos de um AFD e

δ: $Q \times \Sigma \rightarrow 2^Q$, conjunto potência de Q, isto é, todos os sub-conjuntos de Q

$L(M) = \{w \mid \delta(q_0,w) \cap F \neq \emptyset\}$

No exemplo 2:

Estado	Entrada	
	0	1
q_0	$\{q_0,q_3\}$	$\{q_0,q_1\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$

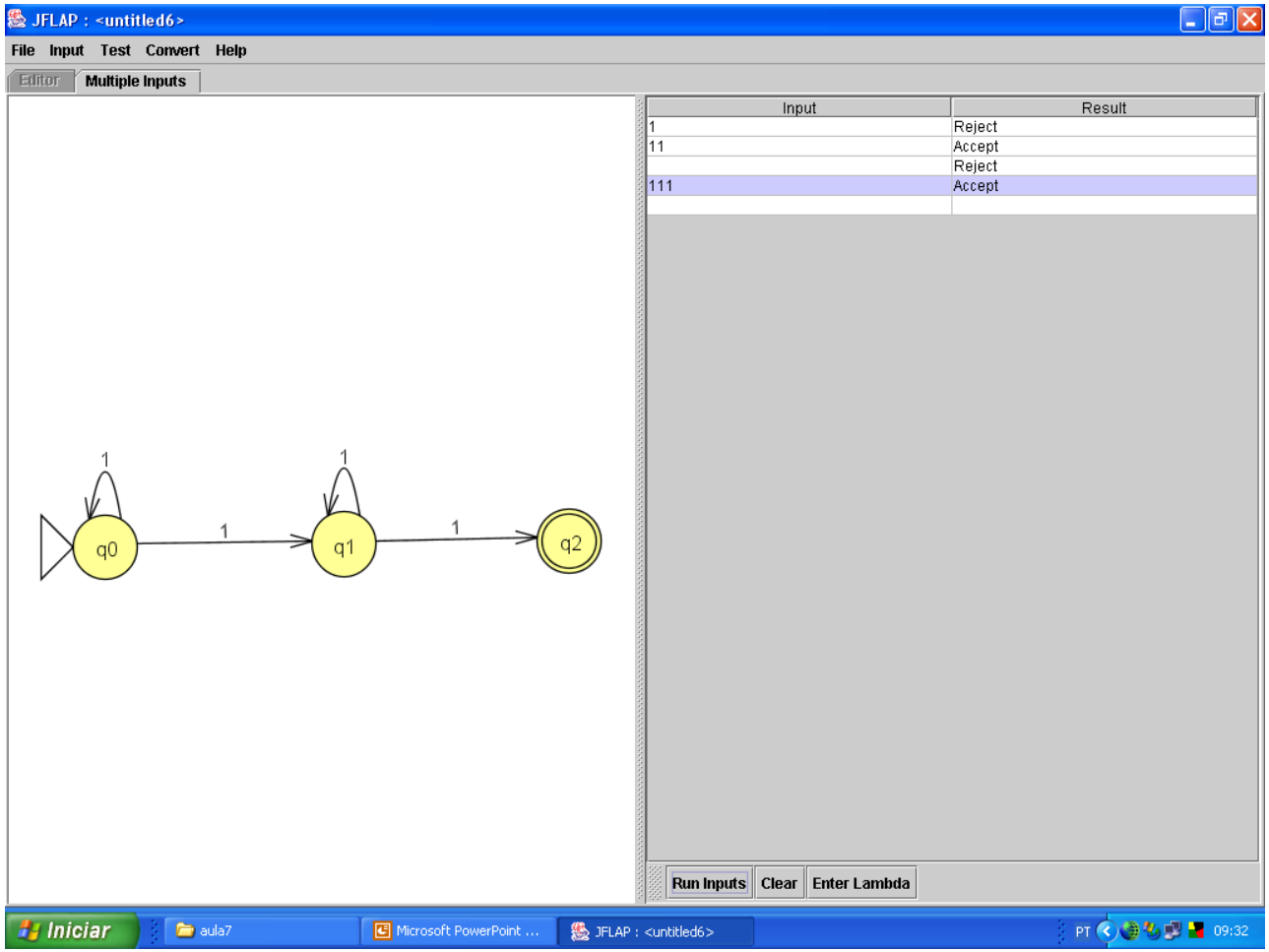
Exemplo 3

Construir um AFND que aceita cadeias $\in \{1,2,3\}^*$ tal que o último símbolo na cadeia tenha aparecido anteriormente. Por exemplo, 121 é aceita; 31312 não é aceita.

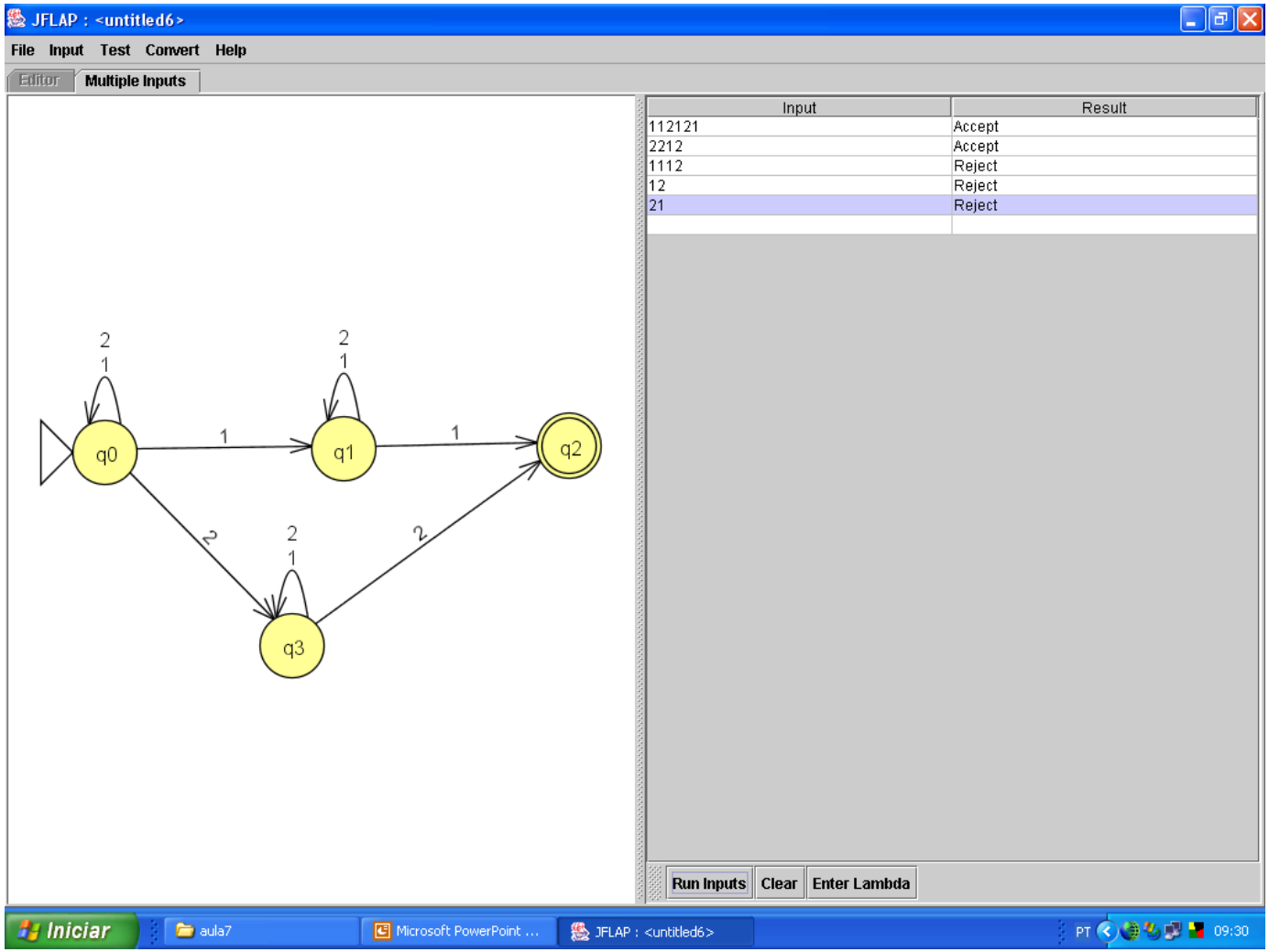
Dica: resolvam para os seguintes vocabulários mais simples antes

1) Construir um AFND que aceita cadeias $\in \{1\}^*$ tal que o último símbolo na cadeia tenha aparecido anteriormente
2) Construir um AFND que aceita cadeias $\in \{1,2\}^*$ tal que o último símbolo na cadeia tenha aparecido anteriormente

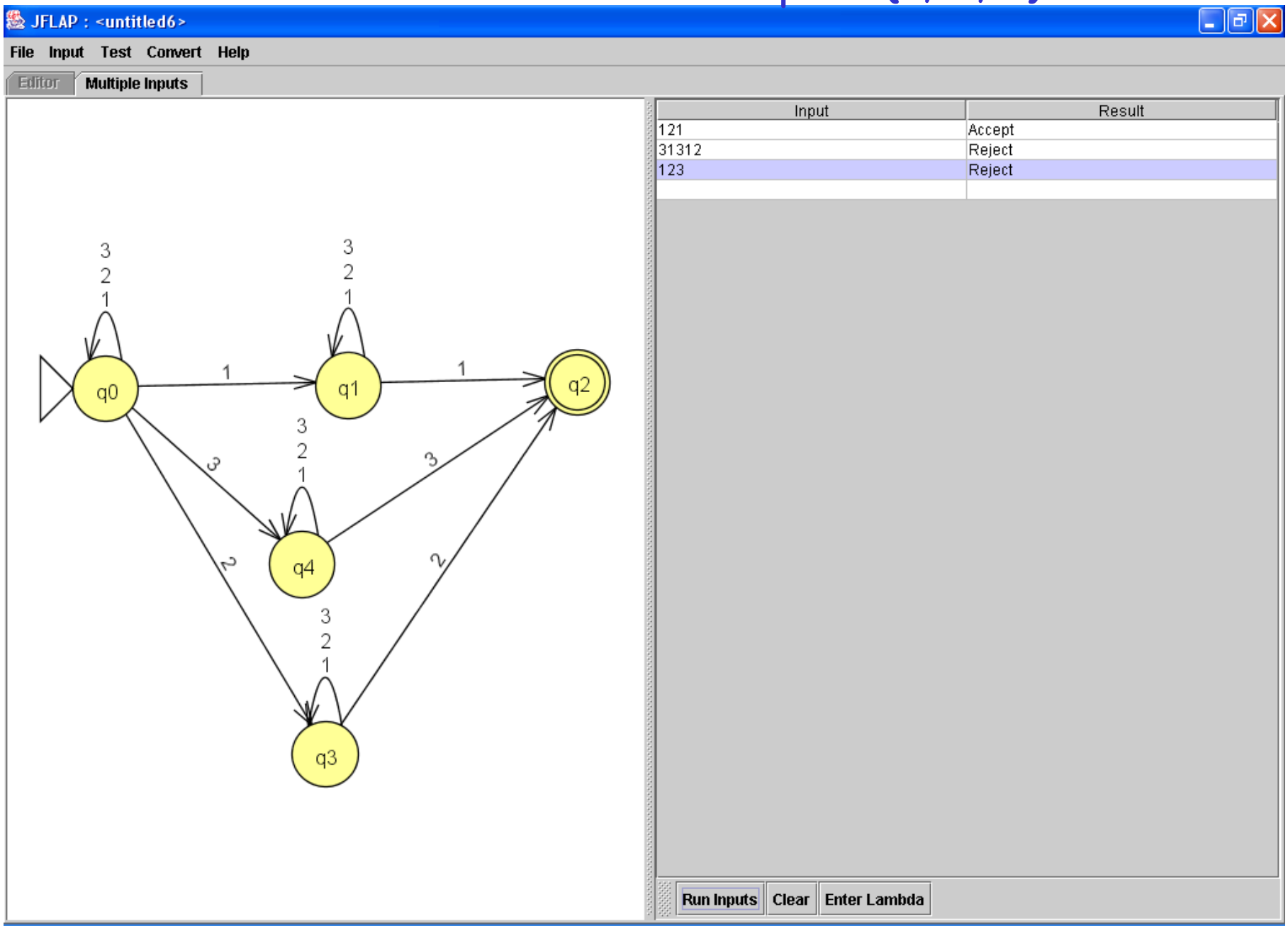
1) Construir um AFND que aceita cadeias $\in \{1\}^*$ tal que o último símbolo na cadeia tenha aparecido anteriormente



2) Construir um AFND que aceita cadeias ∈ {1,2}* tal que o último símbolo na cadeia tenha aparecido anteriormente



Estendendo o vocabulário para {1, 2, 3}:



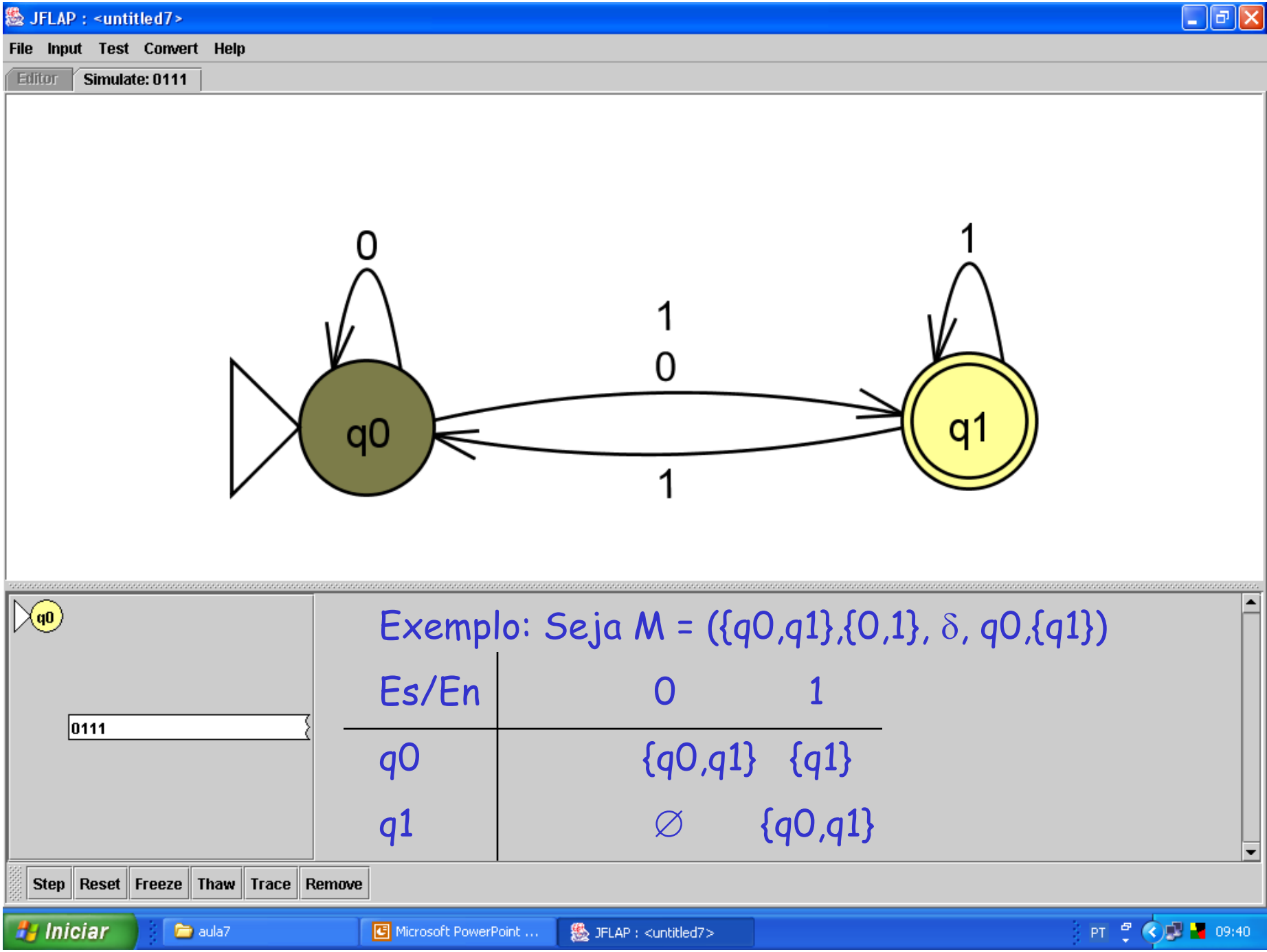
Equivalência entre AFD e AFND

Teorema: Se L é reconhecida por um AFND, então ela é reconhecida por um AFD.

Esse teorema responde a primeira pergunta desses slides.

Vamos propor um método para construir um AFD a partir do AFND. Daí se pode provar que as linguagens reconhecidas por ambos são iguais (veja essa prova na bibliografia).

Esse método é chamado de *construção de subconjuntos*


JFLAP : <untitled7>
File Input Test Convert Help
Editor Simulate: 0111
0
1
0
1
1
q0
q1
q0
0111
Exemplo: Seja M = ({q0,q1},{0,1}, δ, q0,{q1})
Es/En 0 1
q0 {q0,q1} {q1}
q1 ∅ {q0,q1}
Step Reset Freeze Thaw Trace Remove
Iniciar
aula7
Microsoft PowerPoint ...
JFLAP : <untitled7>
PT
09:40

Nós podemos construir um AFD M'= (Q', {0,1}, δ', {qo}, F') aceitando L(M) pela construção chamada de "construção de subconjuntos" (dos estados de AFND). Ela é tal que Σ é o mesmo do AFND e o estado inicial é o conjunto contendo somente o estado inicial do AFND.

- Q' consiste de todos os subconjuntos de {qo,q1}:

  eo e1 e2 e3

  {qo}, {q1}, {qo,q1}, e $\varnothing$.

- $\delta'(\{q_1 \ldots q_n\}, a) = {}_{i=1}\cup^{n} \delta(qi, a)$;

- F' = {p | p $\subseteq$ Q' e p contém ao menos 1 elemento de F }

Assim, no exemplo:

$\delta'(\{q_0\},0) = \{q_0,q_1\}$ $\delta'(\{q_0\},1) = \{q_1\}$

$\delta'(\{q_1\},0) = \varnothing$ $\delta'(\{q_1\},1) = \{q_0,q_1\}$

$\delta'(\{q_0,q_1\},0) = \{q_0,q_1\}$

Pois $\delta(\{q_0,q_1\},0) = \delta(q_0,0) \cup \delta(q_1,0) = \{q_0,q_1\} \cup \varnothing = \{q_0,q_1\}$

$\delta'(\{q_0,q_1\},1) = \{q_0,q_1\}$

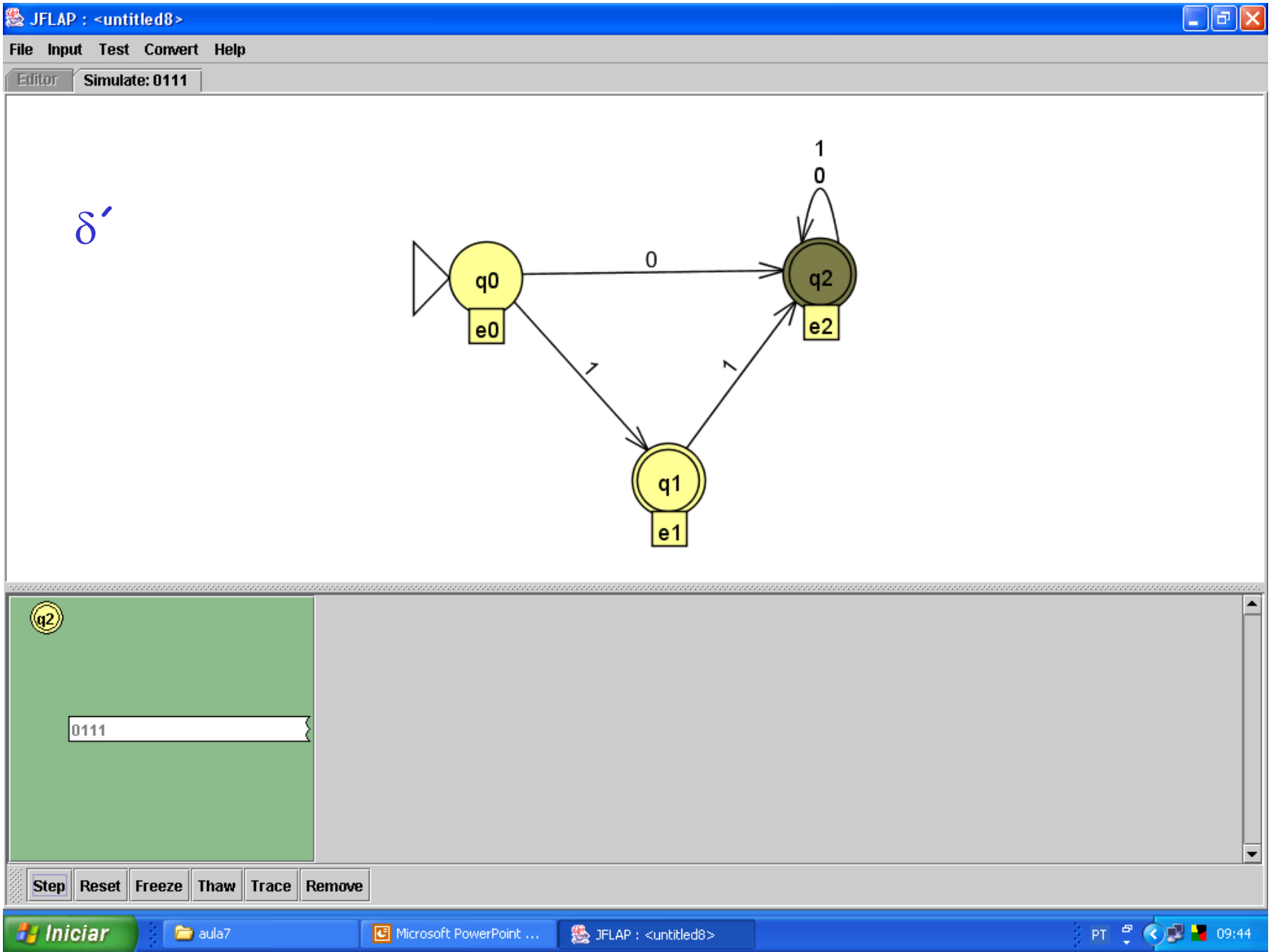
Pois $\delta(\{q_0,q_1\},1) = \delta(q_0,1) \cup \delta(q_1,1) = \{q_1\} \cup \{q_0,q_1\} = \{q_0,q_1\}$

$\delta'(\varnothing,0) = \delta'(\varnothing,1) = \varnothing$

- F' = {{q1},{q0,q1}} isto é, estados onde F antigo estava presente

δ	0	1
$\varnothing$	$\varnothing$	$\varnothing$
$\{q_0\}$	$\{q_0,q_1\}$	$\{q_1\}$
$\{q_1\}$	$\varnothing$	$\{q_0,q_1\}$
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_1\}$

M'= ({eo,e1,e2}, {0,1}, δ', eo, {e1,e2})

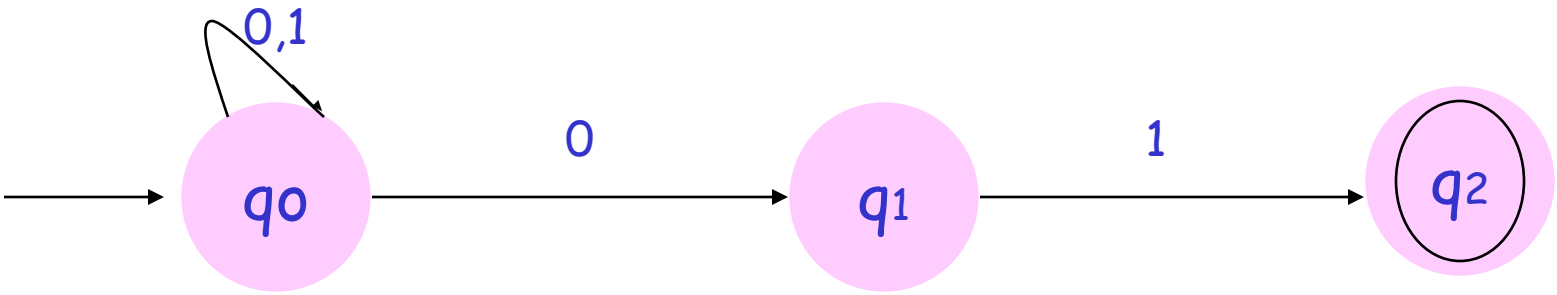


Estados Acessíveis e Não Acessíveis

Embora muitas vezes seja mais fácil construir um AFND para uma LR, o AFD tem na prática quase o mesmo número de estados que o AFND, embora ele tenha mais transições.

No pior caso, o AFD pode ter 2^n estados enquanto que o AFND para a mesma linguagem tem somente n estados.

Ex. seja o AFND que aceita todas as cadeias em {0,1} que terminam em 01.



AFND

	0	1
→ q0	{q0, q1}	{q0}
q1	∅	{q2}
q2	∅	∅

AFD

	0	1
∅	∅	∅
→ {q0}	{q0,q1}	{q0}
*{q1}	∅	{q2}
*{q2}	∅	∅
{q0, q1}	{q0,q1}	{q0,q2}
* {q0,q2}	{q0,q1}	{q0}
*{q1,q2}	∅	{q2}
*{q0,q1,q2}	{q0,q1}	{q0,q2}

Renomeando os estados:

AFND

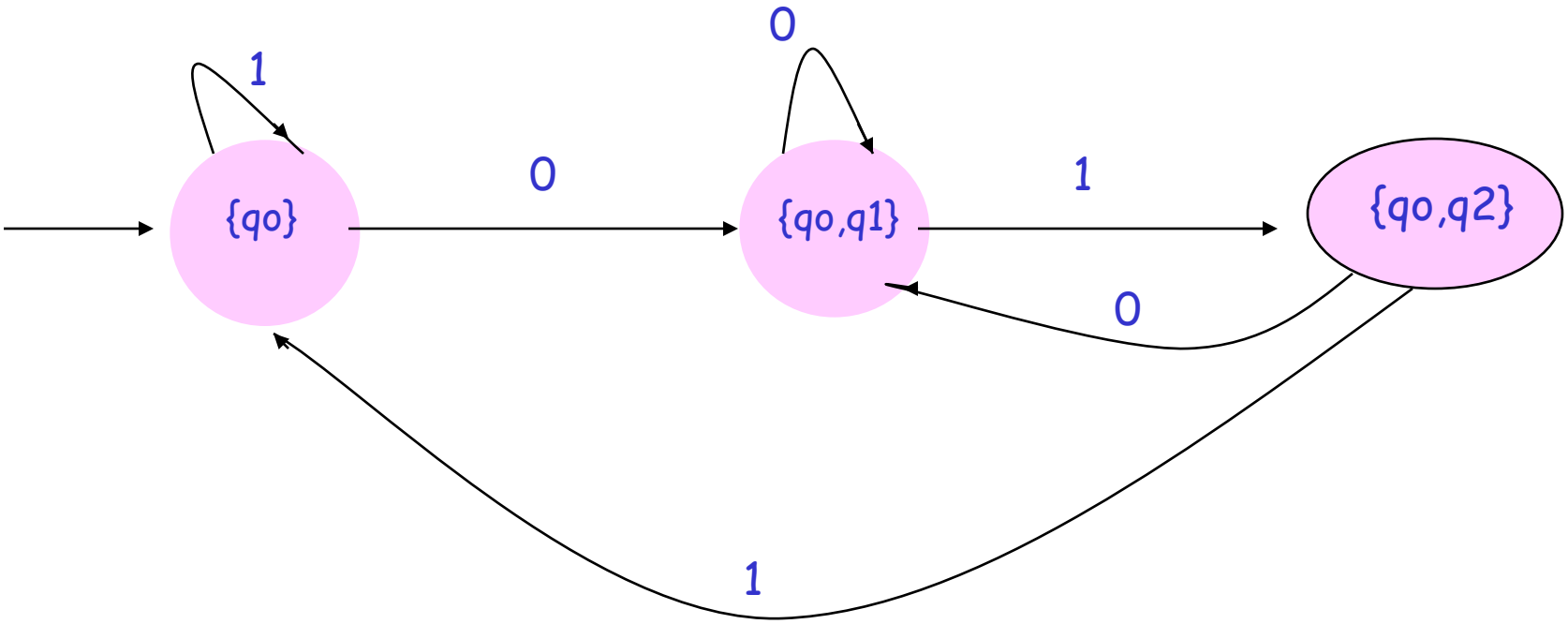
	0	1
→ q0	{q0, q1}	{q0}
q1	∅	{q2}
q2	∅	∅

AFD

	0	1
A	A	A
→ B	E	B
* C	A	D
*D	A	A
E	E	F
*F	E	B
*G	A	D
*H	E	F

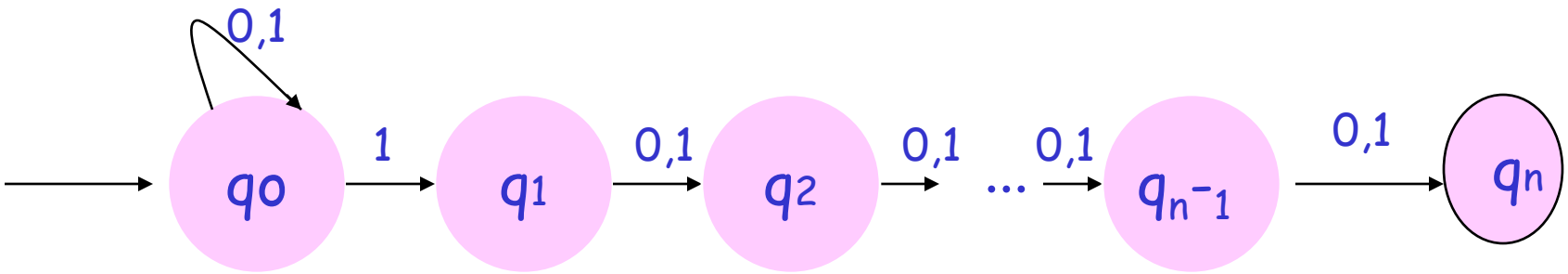
Repare que os únicos estados acessíveis a partir de B são: B, E e F. Os demais são inacessíveis e não precisam constar da tabela.

AFD resultante com n=3 estados



Há casos, no entanto, em que o AFD correspondente não pode ter menos estados do que 2^n, sendo n o número de estados do AFND correspondente.

Verifique para o seguinte AFND:



Aceita cadeias cujo n-ésimo símbolo, a partir da direita, é 1.

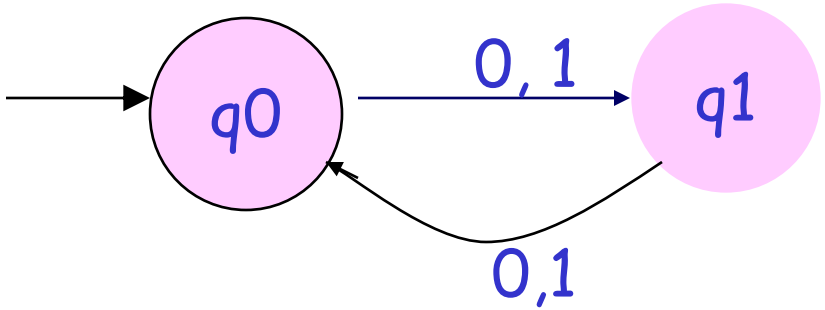
Cuidado com os estados de aceitação!

Muitas vezes, ao construir AFD, impedimos que ele aceite cadeias válidas.

Por outro lado, ao construir AFND, as vezes fazemos com que eles aceitem cadeias que não deveriam aceitar.

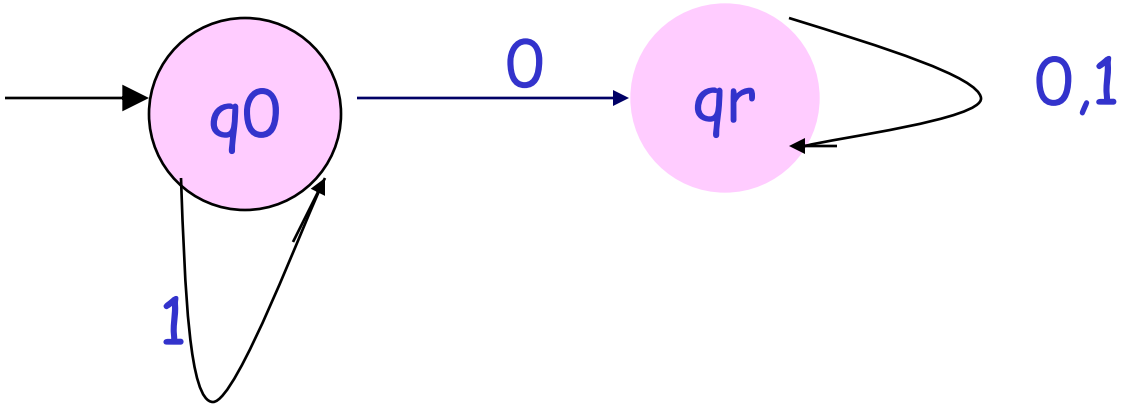
Exemplo

- Seja um AFD A que aceita cadeias sobre {0,1}* de comprimento par



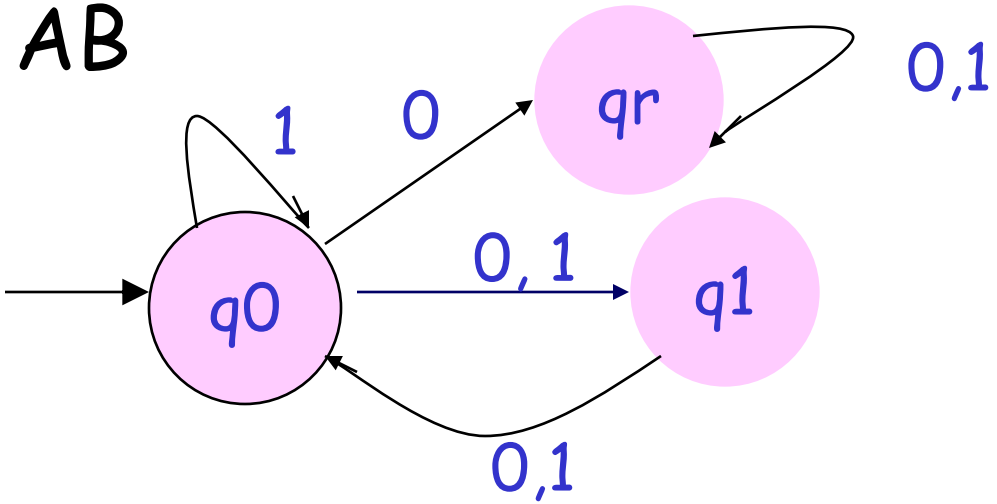
$L(A) = \{(0+1)^{2n} \mid n=0,1,\ldots\}$

- Seja um AFD B que aceita cadeias sobre {0,1}* que não contêm 0´s



L(B)= 1*

- Coloque A e B juntos, formando um AFND AB



- Poderíamos esperar que AB aceitasse a linguagem união das duas anteriores.
- Mas vemos que ela aceita não apenas essa união, mas qualquer cadeia exceto aquelas com número ímpar de 0´s e nenhum 1.
- $L(AB)=\{0,1\}^* - \{0^{2n+1} \mid n \geq 0\}$

Estados de não-aceitação em AFD

- A definição estrita de um AFD EXIGE que todo estado tenha uma transição para cada símbolo de entrada.
- Se seguirmos esta definição muitos exemplos vistos na seção de AFD não eram AFD no senso estrito.
- Porém, temos a facilidade de criar um estado de não-aceitação (erro) para o qual podem ir todas as entradas não necessárias na definição de uma linguagem.

AFD com estado de não-aceitação

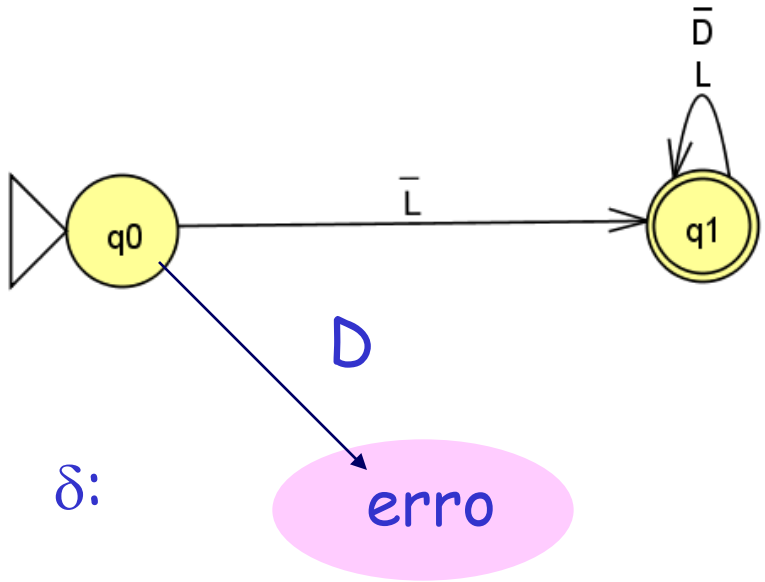
JFLAP : <untitled3>

File Input Test Convert Help

Editor | Multiple Inputs

Input	Result
_LLL	Accept
LDD	Accept
DD	Reject
LLL	Accept
D+D	Reject

$M = (\{q0, q1, erro\}, \{A..Z, a..z, 0..9, \_\}, q0, \delta, \{q1\})$



AF que reconhece identificadores em Pascal

Run Inputs | Clear | Enter Lambda

Iniciar | SCE_521_185 | Microsoft PowerPoint ... | JFLAP : <untitled3> | PT | 10:51

Um AF não é um programa.....

....mas serve como um bom modelo. Vejamos como podemos implementar um AFD:

- de forma direta: escrevendo um código que simule as transições.
- controlado por tabela: simulador universal de AFD

Autômatos Finitos com ε-transições

Vamos permitir transições sobre a cadeia nula, ε. Equivale a permitir que o AFND faça transições espontâneas, sem receber um símbolo de entrada.

Assim como no caso do não-determinismo, esse novo recurso não expande o conjunto de linguagens reconhecidas, mas dá uma certa conveniência de programação adicional.

- Formalmente, só mudamos a função de transição δ:

$$Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$$

$\delta(q,a) = p$ onde a é ε ou um símbolo de Σ

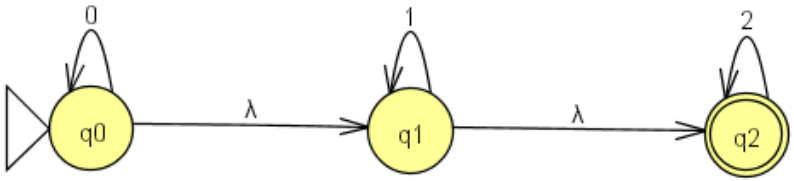
L(M) = (w ∈ {0,1,2}* | w contenha ∀ nro de 0's seguido por ∀ nro de 1's seguido por ∀ número de 2's}

Ex.

JFLAP : <untitled9>

File Input Test Convert Help

Editor | Multiple Inputs

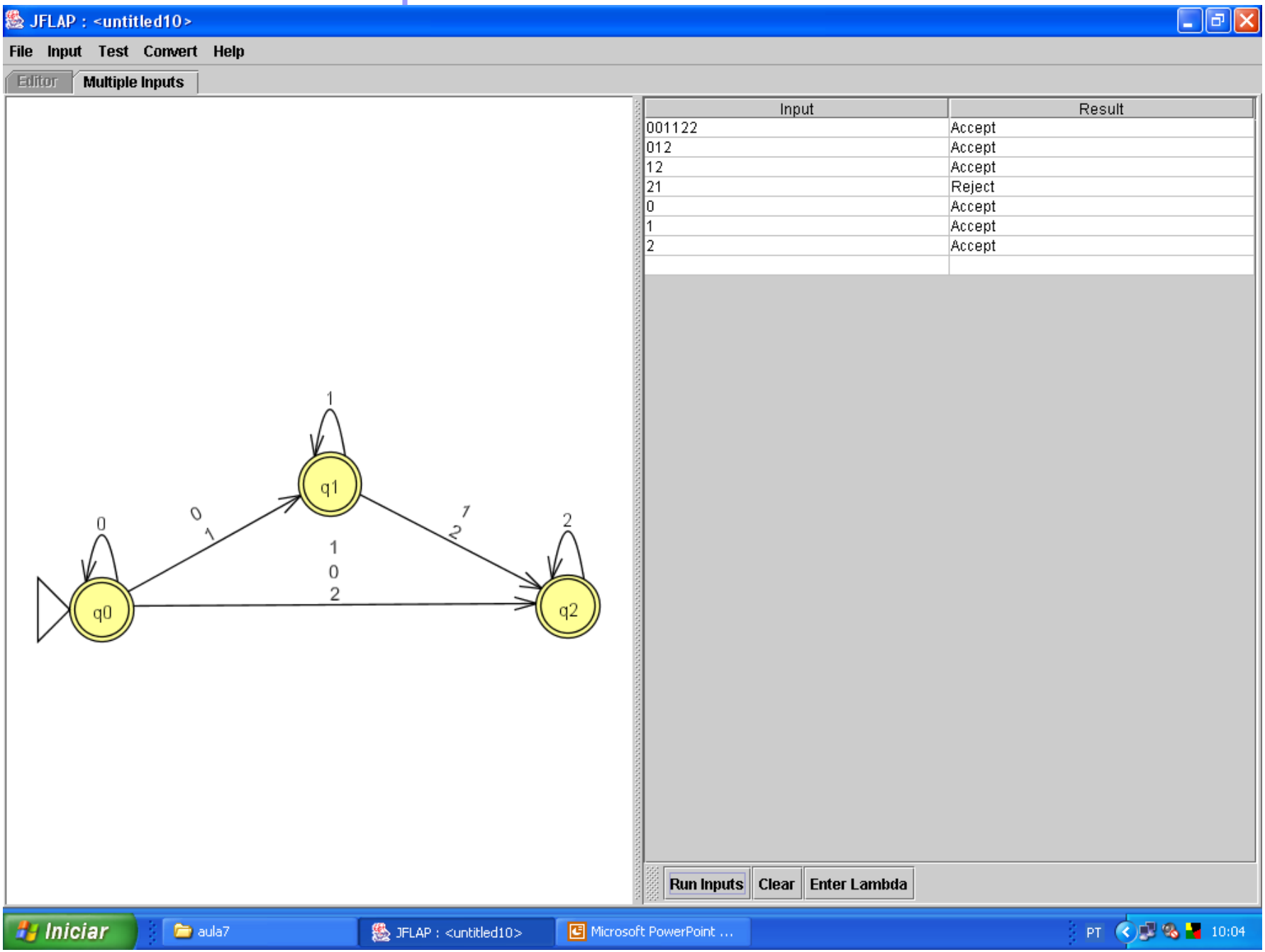


Input	Result
001122	Accept
012	Accept
12	Accept
21	Reject
0	Accept
1	Accept
2	Accept

Run Inputs | Clear | Enter Lambda

Iniciar | aula7 | JFLAP : <untitled9> | Microsoft PowerPoint ... | PT | 10:00

Exemplo anterior com AFND

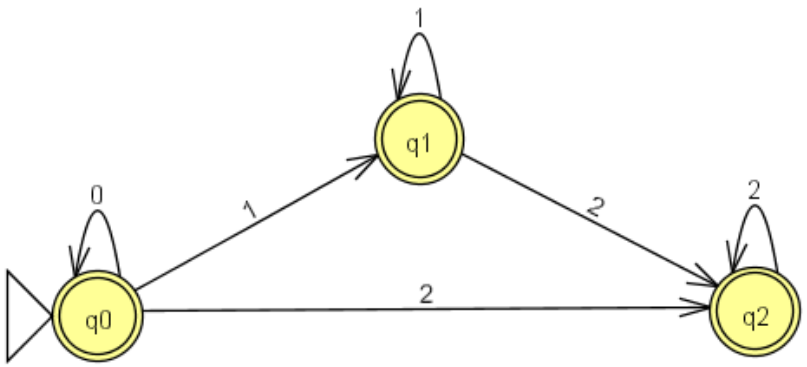


Exemplo anterior com AFD

JFLAP : <untitled10>

File Input Test Convert Help

Editor | Multiple Inputs



Input	Result
001122	Accept
012	Accept
12	Accept
21	Reject
0	Accept
1	Accept
2	Accept

Run Inputs | Clear | Enter Lambda

Iniciar | aula7 | JFLAP : <untitled10> | Microsoft PowerPoint ... | PT | 10:09

Exemplo

Ex.: Considere um ε-AFND que reconhece números decimais que consistem de:

1. um sinal + ou – opcional
2. uma cadeia de dígitos
3. um ponto decimal
4. outra cadeia de dígitos.

As cadeias (2) e (4) podem ser vazias, mas não simultaneamente.

Diagrama de Transições

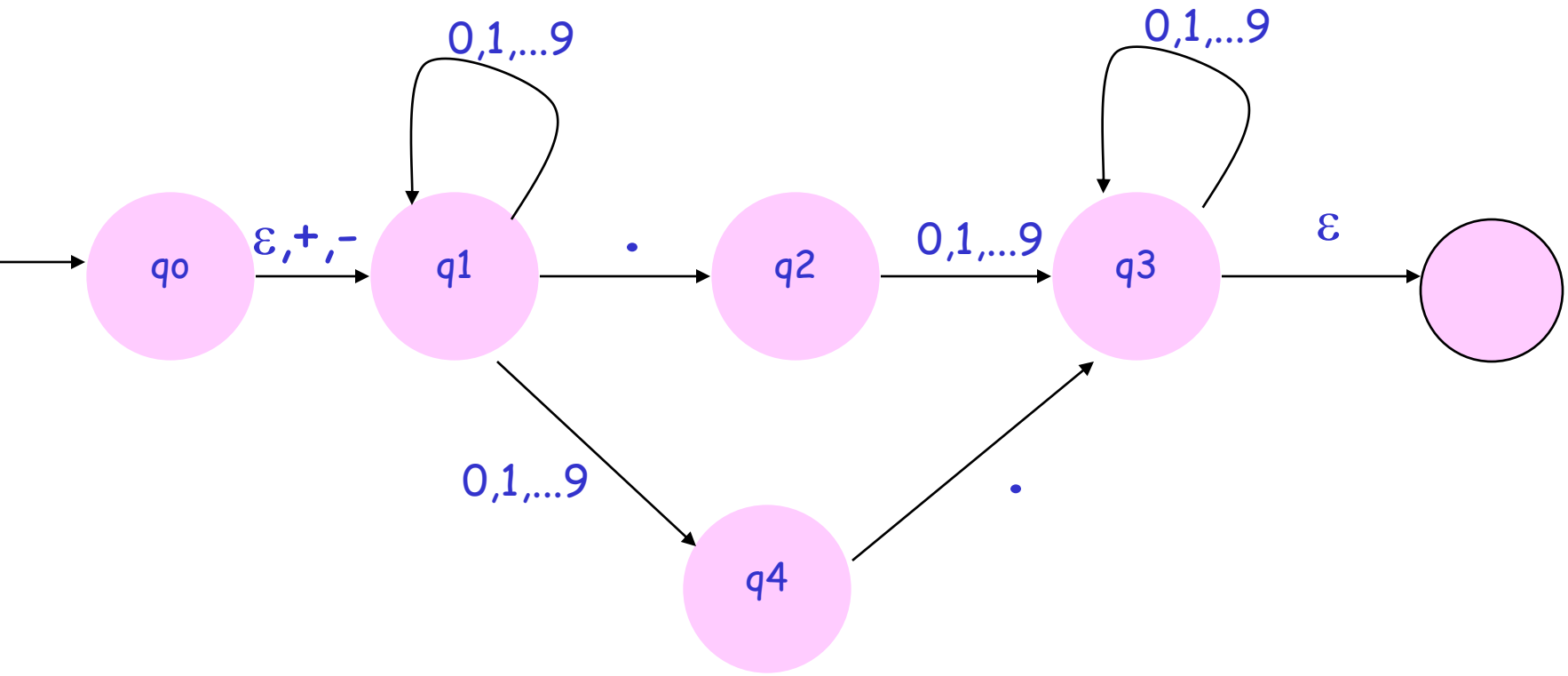


Tabela de Transições

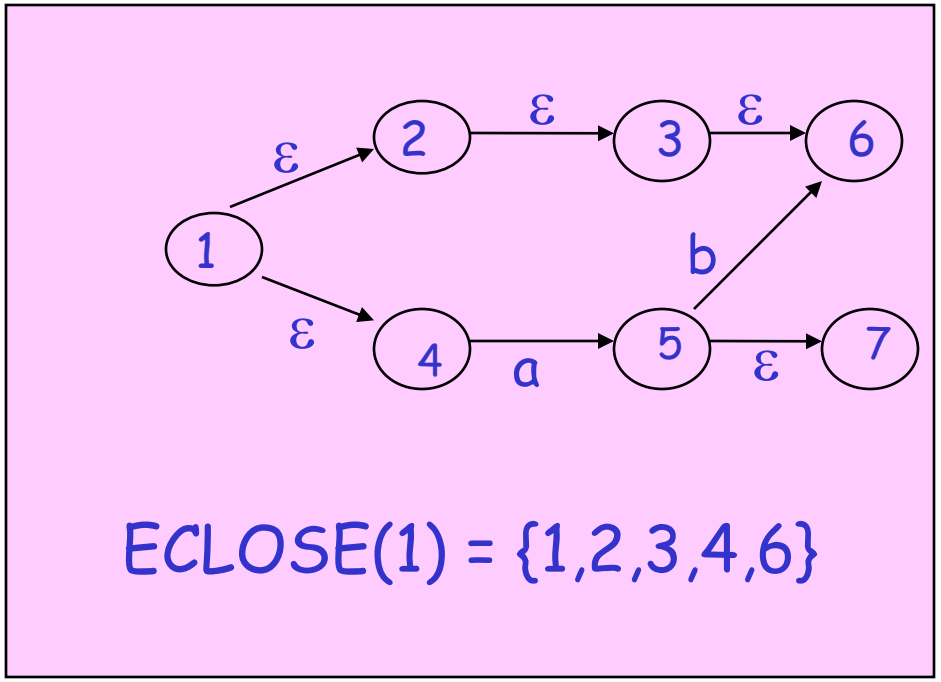
	ε	+,-	.	0,1,...9
→ q0	{q1}	{q1}	∅	∅
q1	∅	∅	{q2}	{q1,q4}
q2	∅	∅	∅	{q3}
q3	{q5}	∅	∅	{q3}
q4	∅	∅	{q3}	∅
q5	∅	∅	∅	∅

ε-Fechamento

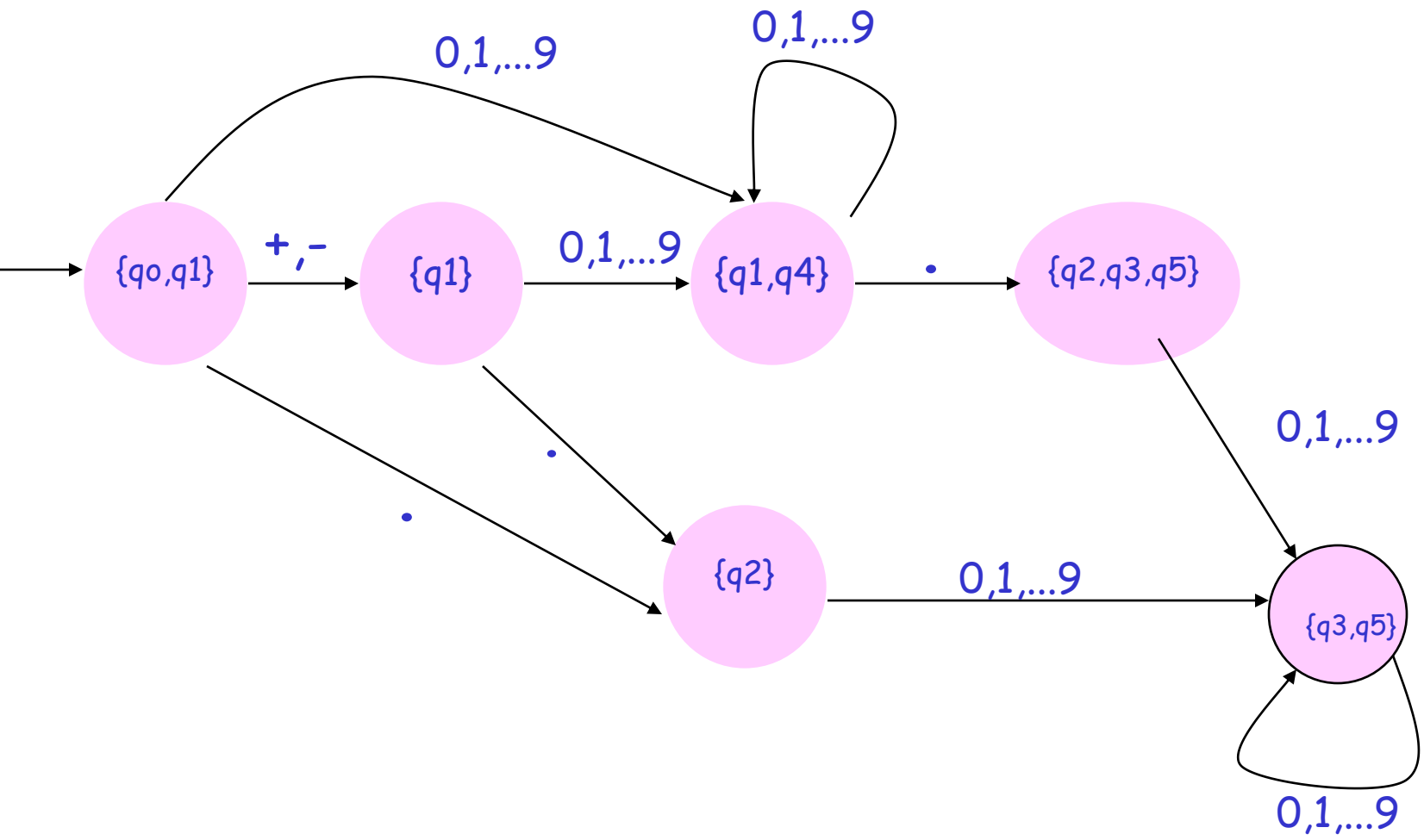
Def.: ε-Fechamento de um estado p, ECLOSE(p):

conjunto dos estados alcançados a partir de p por caminhos rotulados por ε.

(veja definição formal na bibliografia)



O AFD correspondente ao ε-AFND anterior



Teorema:

Uma Linguagem L é aceita por algum ε-AFND se e somente se L é aceita por um AFD (veja prova na bibliografia)

Resumo Parcial

AFD: conjunto finito de estados e conjunto finito de símbolos de entrada. Uma função determina como o estado se altera toda vez que um símbolo é processado.

Linguagem de um AF: Um AF aceita cadeias. Uma cadeia é reconhecida se, começando no estado inicial, as transições levam a um estado de aceitação.

AFND: difere do AFD pelo fato de que pode ter qualquer número de transições (inclusive zero) para os estados seguintes, a partir de um dado estado e de um dado símbolo de entrada.

Construção de subconjuntos: tratando conjuntos de estados de um AFND como estados de um AFD, é possível converter qualquer AFND em um AFD que aceite a mesma linguagem.

Resumo Parcial

ε-transições: podemos estender o AFND permitindo transições para uma entrada vazia, isto é, sem qualquer símbolo de entrada. Esses AFND estendidos podem ser convertidos em AFD que aceitam a mesma linguagem.

(próximo tópico: Expressões Regulares e Linguagens)