

Autómatos Finitos

a^n



Sistemas de Estados Finitos
AF Determinísticos

Relembrando...

Uma representação finita de uma linguagem L qualquer pode ser:

1. Um conjunto finito de cadeias (se L for finita);
2. Uma expressão de um conjunto infinito de cadeias;
3. Uma gramática (que gera L);
4. Um autômato (que reconhece L)

Autômatos Finitos - AF

- São reconhecedores de LINGUAGENS REGULARES.
- São o tipo **mais simples** de Máquina de Turing onde:
 - a fita de entrada é unidirecional: a cadeia é lida da esquerda para a direita, um símbolo por vez;
 - a fita é só de leitura: não é possível gravar símbolos. Isso significa que a única memória utilizada é aquela gasta para armazenar a cadeia de entrada.
- As características acima garantem que o tempo necessário para reconhecer uma cadeia w ; $|w|=n$; é $O(n)$.

Def. Autômato Finito Determinístico - AFD

Um AF Determinístico M é definido como uma quintupla da forma:

$$M=(E, V, f, q_0, F) \text{ onde:}$$

- E é um conjunto finito não vazio de *estados* do autômato finito.
- V é o *alfabeto de entrada* do autômato e corresponde a um conjunto finito não vazio dos *símbolos de entrada* ou *átomos* indivisíveis que compõem a cadeia de entrada submetida ao autômato para aceitação.
- f é uma *função de transição* de estados do autômato e seu papel é o de indicar as transições possíveis em cada configuração do autômato.

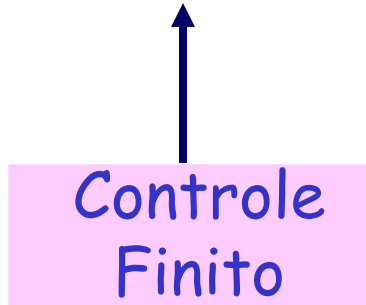
$$f: E \times (V \cup \{\lambda\}) \rightarrow E$$

ou seja, para cada par (*estado*, *símbolo de entrada*), determina o *novo estado* do autômato. $f(p,a) = q \rightarrow$ **determinismo**

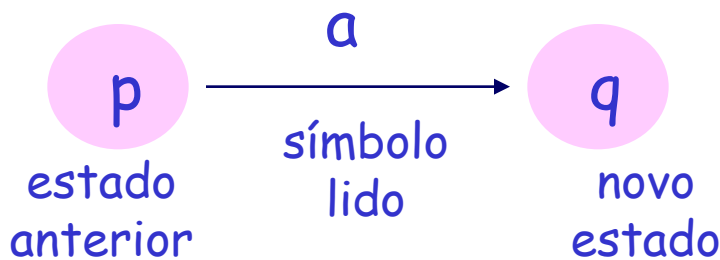
- $q_0 \in E$, é o *estado inicial* do autômato finito.
- $F \subseteq E$ contém todos os *estados de aceitação* ou *estados finais* do AF. São os estados em que o autômato deve terminar o reconhecimento das cadeias de entrada que pertencem à linguagem que o autômato define.

Esquema da Máquina e Representação usando grafos

Esquema:



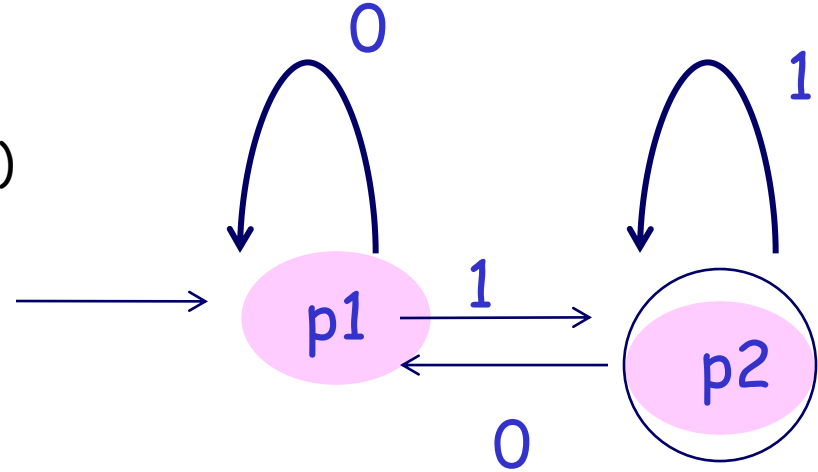
Fita com a
sequência
de símbolos
de V



Exemplo

Exemplo 1:

$M = (\{p1, p2\}, \{0, 1\}, f, p1, \{p2\})$
 $f = (p1, 0) = p1$
 $(p1, 1) = p2$
 $(p2, 1) = p2$
 $(p2, 0) = p1$



- Linguagem reconhecida $L(M)=?$

cadeias de dígitos binários terminadas
obrigatoriamente por um dígito 1: $(0+1)^*1$

- Já que $L(M)$ é regular, dê uma GR que a gere

Pela expressão $(0+1)^*1$:

$S \rightarrow 0S \mid 1S \mid 1$

Pelo AFD:

$S \rightarrow 0S \mid 1A$

$A \rightarrow 0S \mid 1A \mid \lambda$

Tabela de Transição de Estados

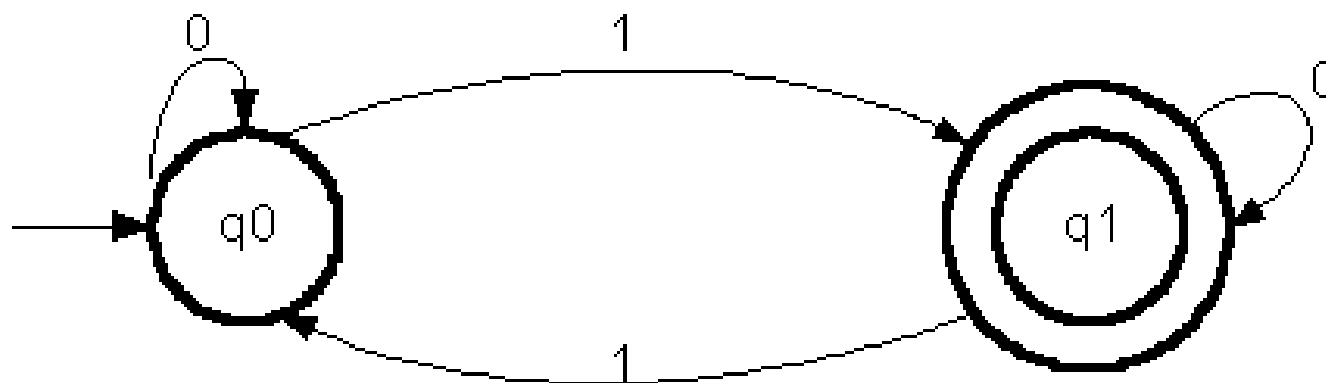
- Forma alternativa para representar Afs
- Exemplo 1:

E\V	0	1
p1	p1	p2
p2 *	p1	p2

estado inicial: →

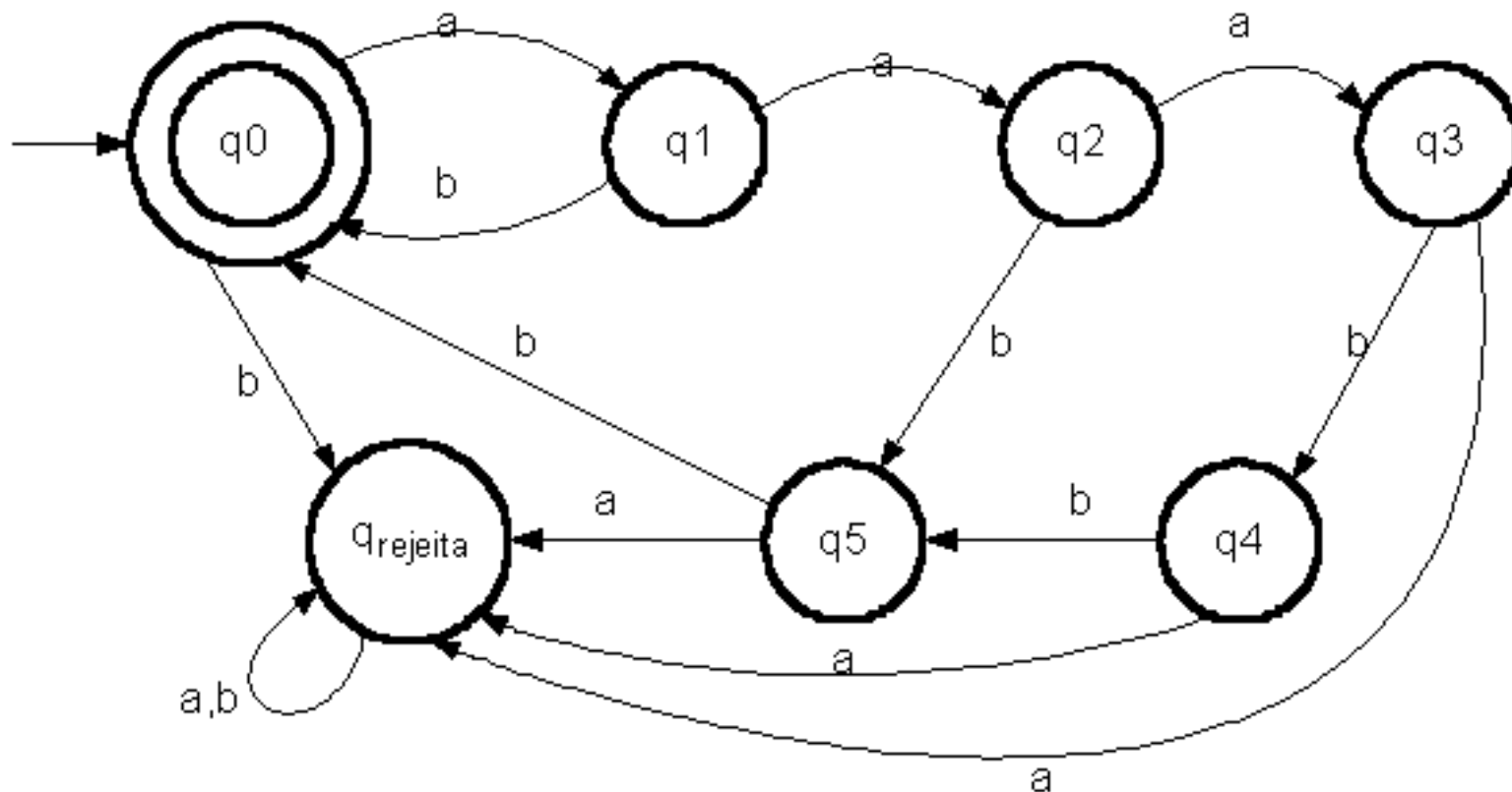
estados finais: *

Exemplo 2:



$L(AF2) =$ conjunto de cadeias de 0's e 1's com número ímpar de 1's

Exemplo 3: Construir um AF M3 para reconhecer as cadeias formadas por um número arbitrário de repetições de 1 a 3 a's seguidos pelo mesmo número de b's. Isto é:
 $L(M3) = (ab|aabb|aaabbb)^*$



Exemplo 4

Seja

$L = \{w \mid w \text{ é da forma } x01y \text{ para algumas cadeias } x \text{ e } y \text{ que consistem somente de } 0\text{'s e } 1\text{'s}\}$ ou

$L = \{ x01y \mid x \text{ e } y \in \{0,1\}^* \}$

Então $01, 11010$ e $100011 \in L$ e

$\lambda, 0$ e $111000 \notin L$.

Vejam os: Seja A o autômato.

$\Sigma = \{0,1\}$. Seja $q_0 \in Q$ o estado inicial. Para decidir se 01 é uma subcadeia da entrada, A precisa lembrar:

1. Ele já viu 01? Se sim, ele aceita toda sequência de caracteres adicionais; i.e, de agora em diante ele estará sempre num estado final (de aceitação).
2. Ele nunca viu 01, mas o último caracter foi 0; assim, se agora ele vir o 1, terá visto 01 e poderá aceitar tudo que vier daqui por diante.
3. Ele nunca viu 01, mas ou ele acabou de iniciar ou acabou de encontrar 1; nesse caso, A não pode aceitar até ver primeiro um 0 seguido de 1.

Cada uma das condições acima pode ser representada por um estado. A condição (3) é o estado inicial q_0 . Ficamos nesse estado até aparecer um 0. Logo $\delta(q_0, 1) = q_0$.

Mas se estamos em q_0 e vemos um 0, estamos na condição (2). Seja q_2 esse estado. Assim, $\delta(q_0, 0) = q_2$

Se, em q_2 , vemos um 0, ainda não encontramos 01, mas estamos na mesma situação de ter o 0 e esperar o 1; logo, ficamos no mesmo estado: $\delta(q_2, 0) = q_2$.

Se, no entanto, em q_2 , vemos 1, então teremos a subcadeia 01 e podemos ir para um estado final, que corresponde à situação (1); chamaremos de q_1 . Logo, $\delta(q_2, 1) = q_1$.

Em q_1 , já vimos a sequência 01, então não importa o que aparecer, ainda estaremos na situação de já ter visto 01. Isto é, $\delta(q_1, 0) = \delta(q_1, 1) = q_1$.

Dessa forma, $Q = \{q_0, q_1, q_2\}$. q_0 é o estado inicial e $F = \{q_1\}$:

$$A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\}),$$

onde δ é a função de transição descrita anteriormente.

Linguagem aceita por um AF M

- Uma cadeia x é aceita pelo AFD $M = (Q, \Sigma, \delta, q_0, F)$ se existe uma sequência de transições a partir de q_0 , de modo que após ler o último símbolo de x , o estado atual é algum $p \in F$.
- Def 1: Uma linguagem aceita por um AFD é uma **Linguagem Regular**
- Def 2: Dois AF M_1 e M_2 são equivalentes se e somente se $L(M_1) = L(M_2)$

Exercício

Fazer um AFD M tal que

$L(M) = \{w \mid w \in \{0,1\}^* \text{ e possui um número par de ocorrências de } 0\text{'s e de } 1\text{'s}\}$

<http://www.jflap.org/>

Exemplo 1

Fazer um AFD M que aceita $L(M) = \{w \mid w \text{ possui um nro par de } 0\text{'s e de } 1\text{'s}\}$

Diagrama de Transição de Estados

```
graph LR; q0((q0)) -- 1 --> q1((q1)); q1 -- 1 --> q0; q0 -- 0 --> q2((q2)); q2 -- 0 --> q0; q1 -- 0 --> q3((q3)); q3 -- 0 --> q1; q2 -- 1 --> q3; q3 -- 1 --> q2;
```

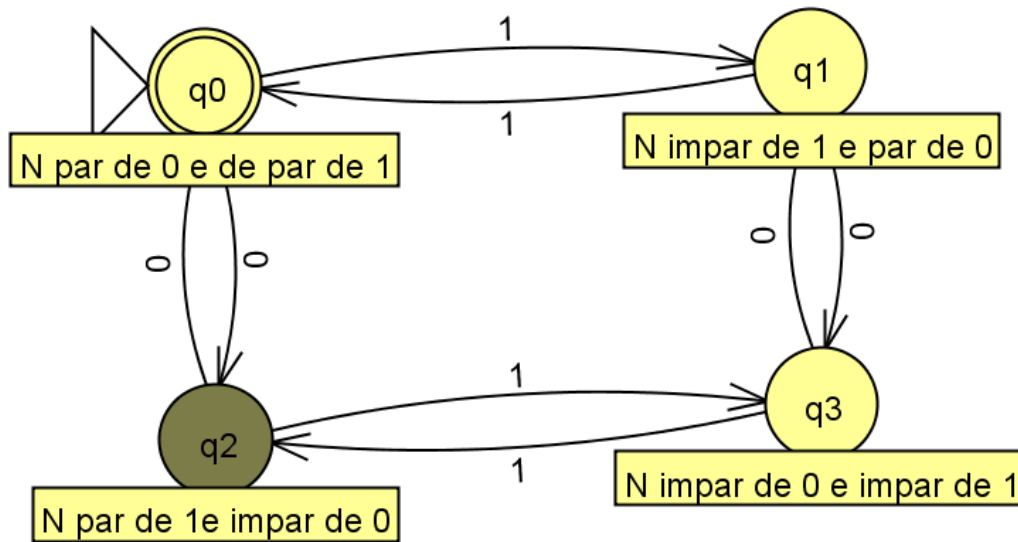
Cadeia aceita:
configuração final
VERDE

Traceback

▶ (q0)	110101
● (q1)	110101
▶ (q0)	110101
● (q2)	110101
● (q3)	110101
● (q1)	110101
▶ (q0)	110101

Step Reset Freeze Thaw Trace Remove

Windows Taskbar: Iniciar, SCE_521_185, automatos.ppt, JFLAP 4.0 Beta Ve..., JFLAP : <untitled1>, Traceback, PT, 15:44



q2

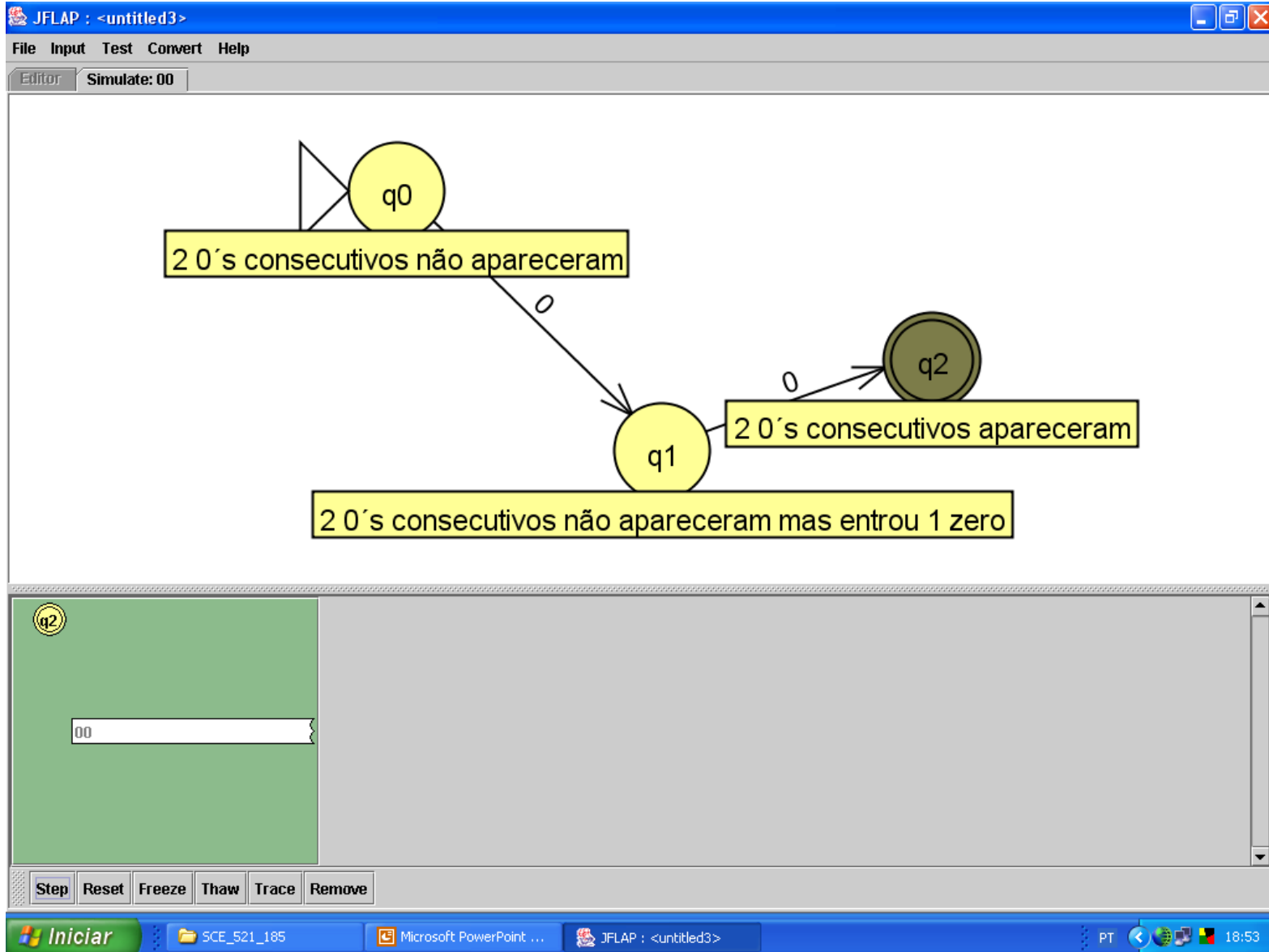
110

Cadeia não aceita:
configuração final
ROSA

Fazer um AFD M que aceita

$L(M) = \{w \in \{0,1\}^* \mid w \text{ possui pelo menos dois } 0\text{'s consecutivos}\}$

Garantindo
a
restrição...



Completando o AFD M e reconhecendo uma cadeia de L(M)

$$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$$

JFLAP : <untitled3>

File Input Test Convert Help

Editor Simulate: 1110010101

δ :

```
graph LR; q0((q0)) -- 1 --> q0; q0 -- 0 --> q1((q1)); q1 -- 1 --> q0; q1 -- 0 --> q2(((q2))); q2 -- 0 --> q2; q2 -- 1 --> q2;
```

q2

1110010101

Fim de uma simulação, entrada em cinza

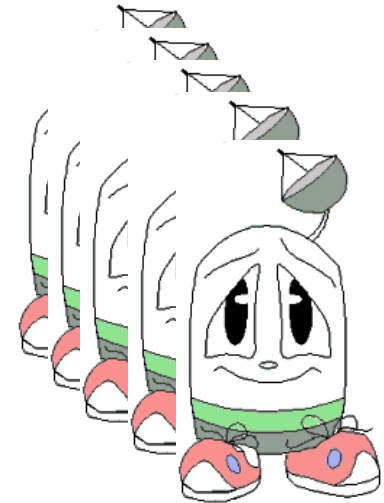
Step Reset Freeze Thaw Trace Remove

Iniciar SCE_521_185 Microsoft PowerPoint ... JFLAP : <untitled3> PT 19:00

Exercícios

Construa gramáticas regulares para as linguagens dos exemplos anteriores de AFs.

Autômatos Finitos



AF Não-determinísticos
Equivalência entre AFND e AFD

AF NÃO-Determinístico (AFND)

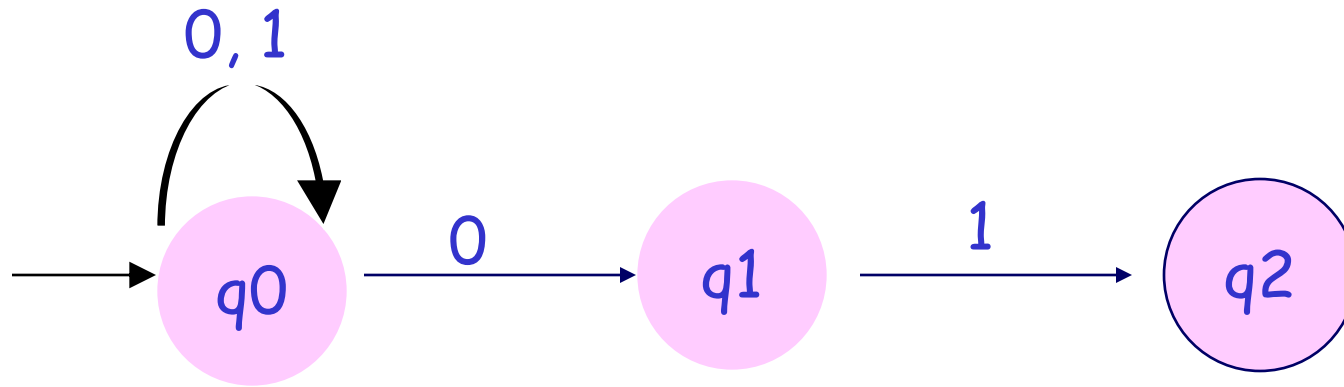
- Consideremos uma modificação no modelo do AFD para permitir
 - zero, uma ou mais transições de um estado sobre o **MESMO** símbolo de entrada.
 - Ou visto de outra forma:
 - a função toma um estado e uma entrada e devolve zero, um ou mais estados.
- Esse modelo é chamado AFND.

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

Exemplo 1

Seja um autômato que aceita a linguagem das cadeias de 0's e 1's que terminam em 01: $(0+1)^*01$.



Seja a cadeia 00101

Quando está em q_0 e o símbolo lido é 0 ele tem a opção de:

- continuar em q_0 no caso do fim da cadeia não estar próximo OU
- ir para q_1 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.

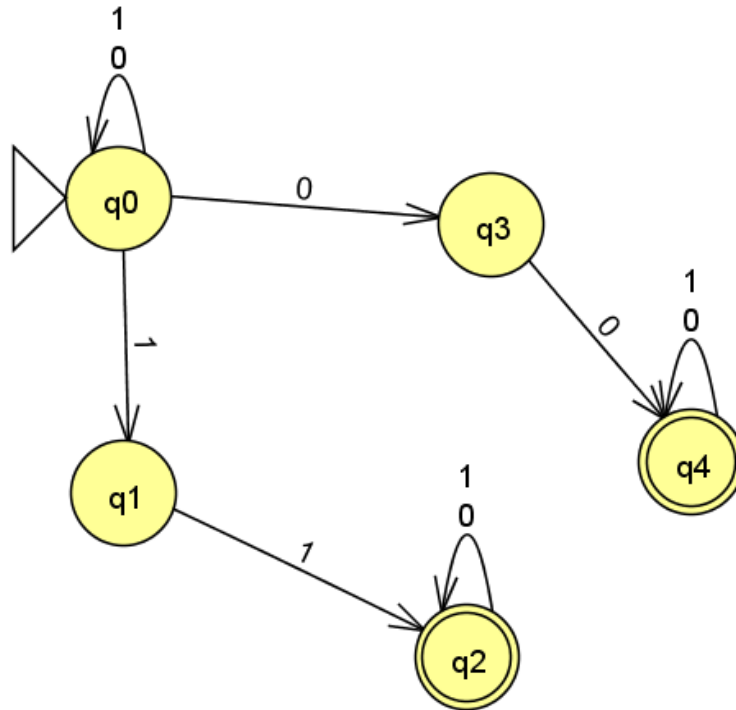
AFND

- Uma cadeia de entrada $a_1a_2\dots a_n$ 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 2

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

Ex 2



Input	Result
110010101	Accept
11010	Accept
00101	Accept

Run Inputs Clear Enter Lambda

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

$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_0, \{q_2, q_4\})$

Definição Formal de um AFND

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

$\delta: Q \times \Sigma \rightarrow 2^Q$, conjunto potência de Q , isto é, todos os subconjuntos 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	$\{q_4\}$	$\{q_4\}$

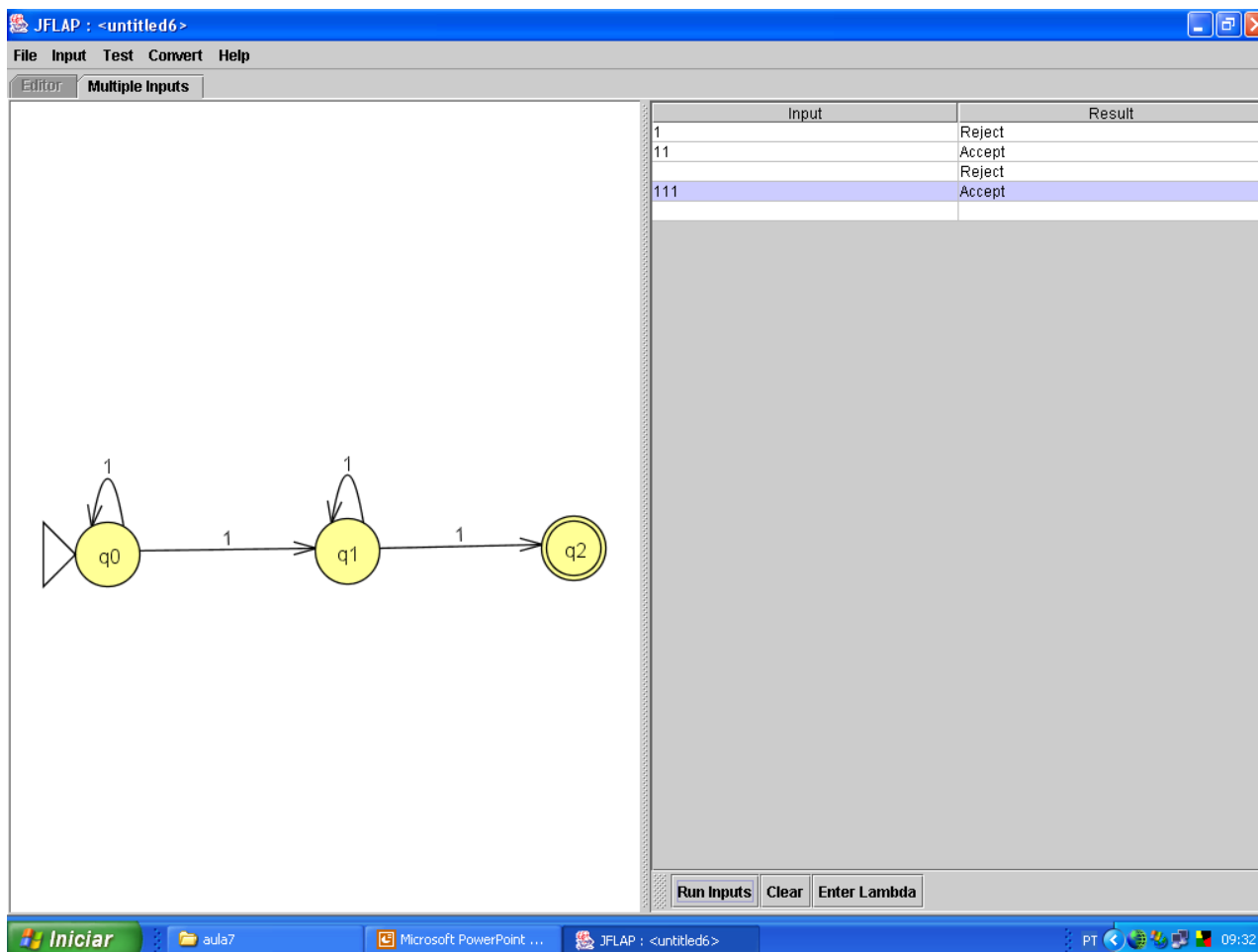
Exemplo 3

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.**

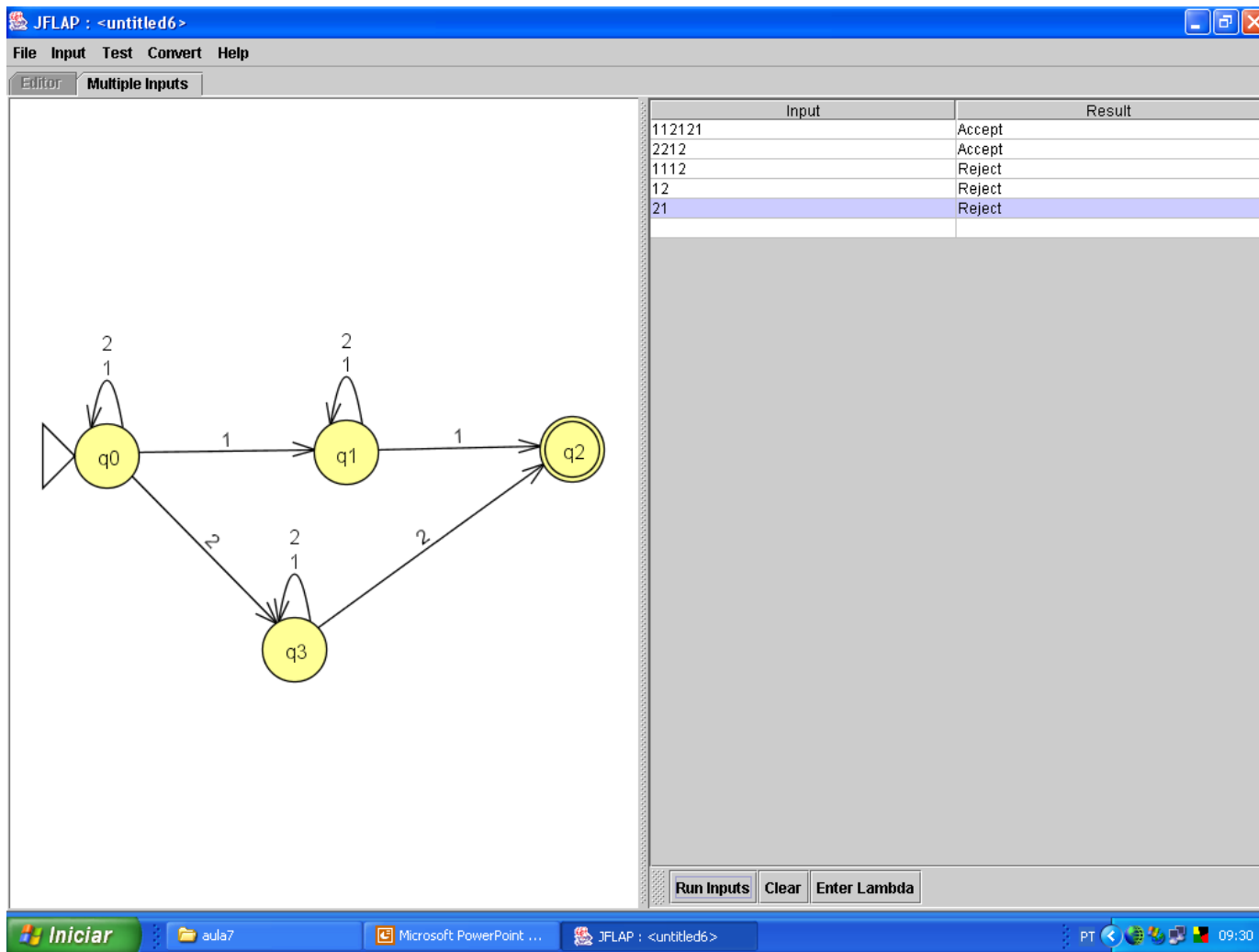
Dica: resolvam para os seguintes vocabulários mais simples antes

- 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 $\in \{1,2\}^*$ tal que o último símbolo na cadeia tenha aparecido anteriormente



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

JFLAP : <untitled6>

File Input Test Convert Help

Editor Multiple Inputs

```
graph LR; q0((q0)) -- 1 --> q1((q1)); q0 -- 2 --> q3((q3)); q0 -- 3 --> q4((q4)); q1 -- 1 --> q2(((q2))); q1 -- 2 --> q4; q1 -- 3 --> q4; q2 -- 1 --> q1; q2 -- 2 --> q3; q2 -- 3 --> q4; q3 -- 1 --> q4; q3 -- 2 --> q0; q3 -- 3 --> q0; q4 -- 1 --> q0; q4 -- 2 --> q0; q4 -- 3 --> q0;
```

Input	Result
121	Accept
31312	Reject
123	Reject

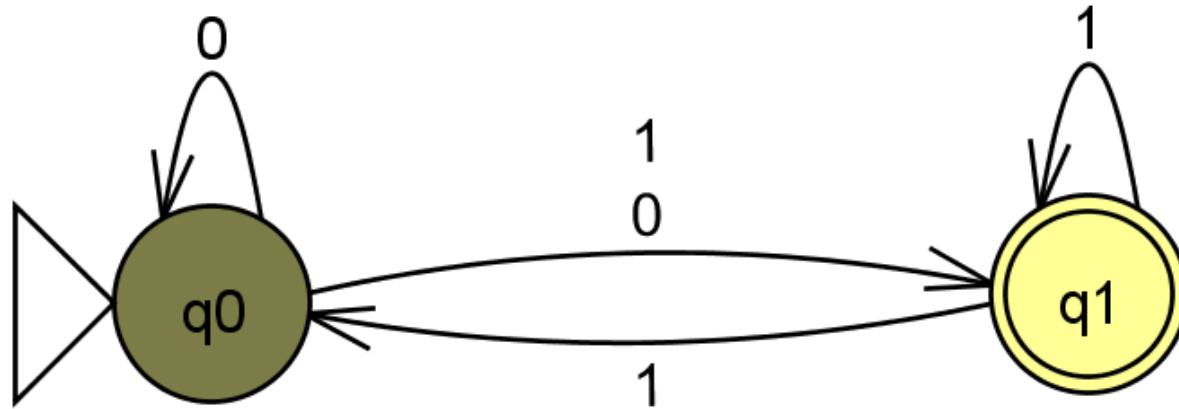
Run Inputs Clear Enter Lambda

Equivalência entre AFD e AFND

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

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*



q0

0111

Exemplo: Seja $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$

Es/En	0	1
q0	{q0, q1}	{q1}
q1	∅	{q0, q1}

Nós podemos construir um AFD $M' = (Q', \{0,1\}, \delta', \{q_0\}, F')$ 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 $\{q_0, q_1\}$:
$$\begin{array}{cccc} e_0 & e_1 & e_2 & e_3 \\ \{q_0\}, & \{q_1\}, & \{q_0, q_1\}, & e \emptyset. \end{array}$$
- $\delta'(\{q_1 \dots q_n\}, a) = \bigcup_{i=1}^n \delta(q_i, a)$;
- $F' = \{p \mid p \subseteq Q' \text{ e } p \text{ contém ao menos 1 elemento de } F\}$

Assim, no exemplo:

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

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

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

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

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

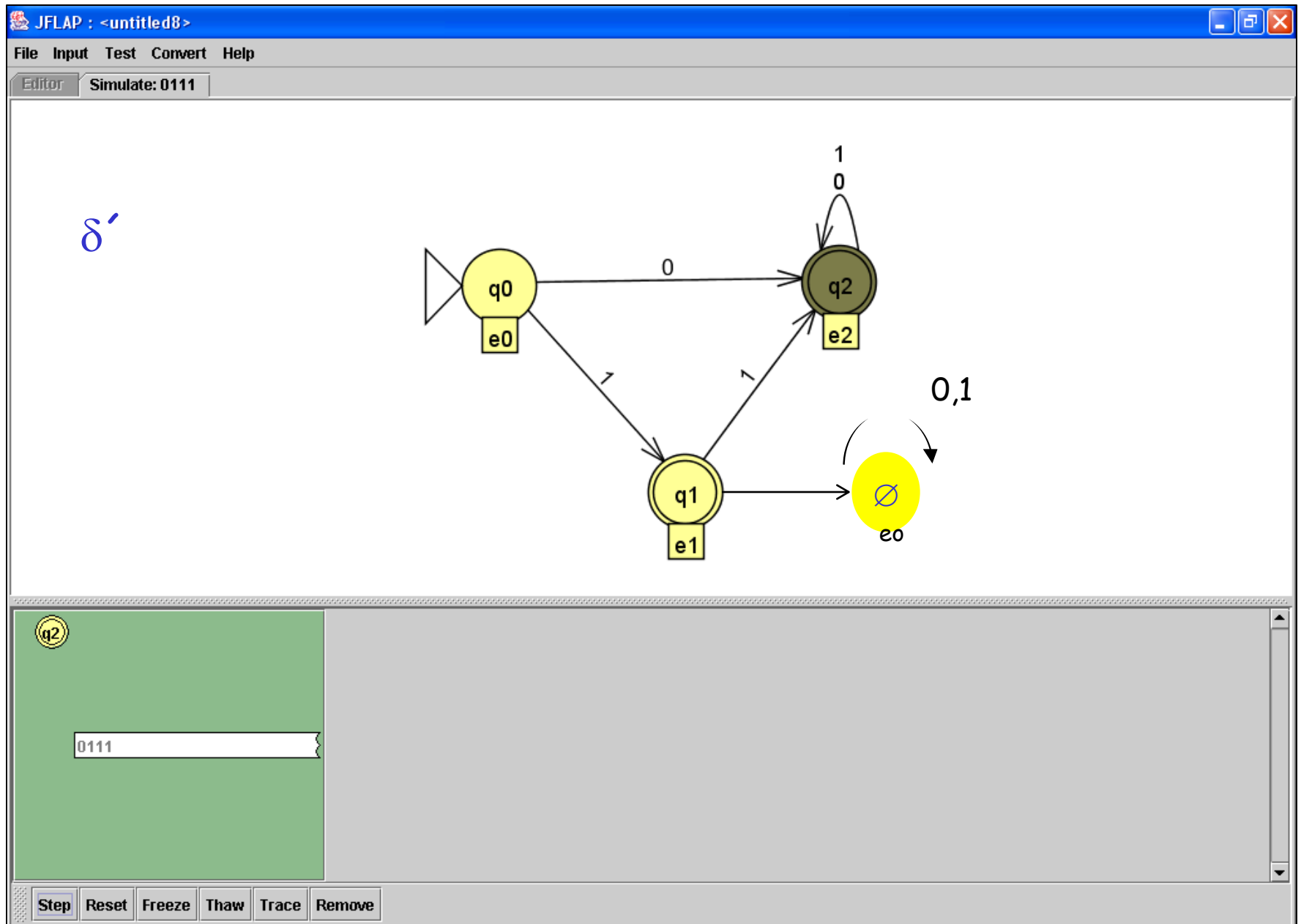
$$\text{Pois } \delta(\{q_0, q_1\},1) = \delta(q_0,1) \cup \delta(q_1,1) = \{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$$

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

• $F' = \{\{q_1\}, \{q_0, q_1\}\}$ isto é, estados onde F antigo estava presente

δ'	0	1
\emptyset (e3)	\emptyset (e3)	\emptyset (e3) (estado de rejeição)
→ $\{q_0\}$ (e0)	$\{q_0, q_1\}$ (e2)	$\{q_1\}$ (e1)
$\{q_1\}^*$ (e1)	\emptyset (e3)	$\{q_0, q_1\}$ (e2)
$\{q_0, q_1\}^*$ (e2)	$\{q_0, q_1\}$ (e2)	$\{q_0, q_1\}$ (e2)

$$M' = (\{e_0, e_1, e_2\}, \{0, 1\}, \delta', e_0, \{e_1, e_2\})$$

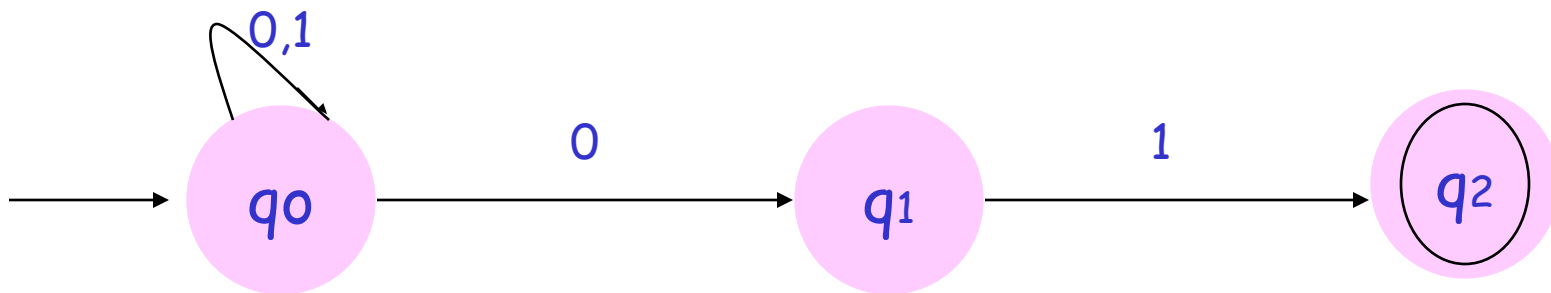


Estados Acessíveis e Não Acessíveis

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 com 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	∅	∅

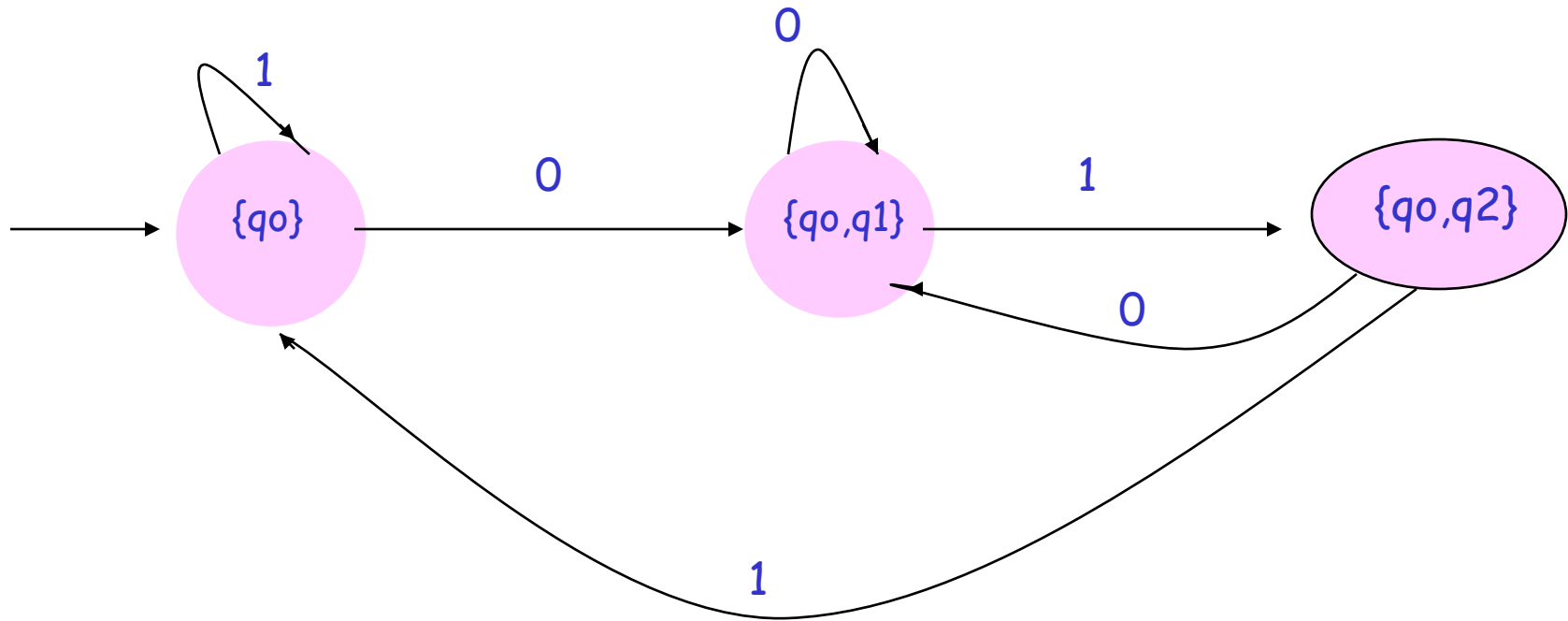
AFD

	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.

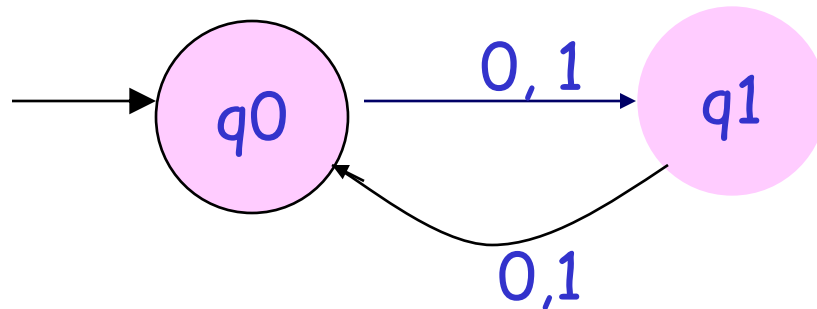
Cuidado com os estados de aceitação!

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.

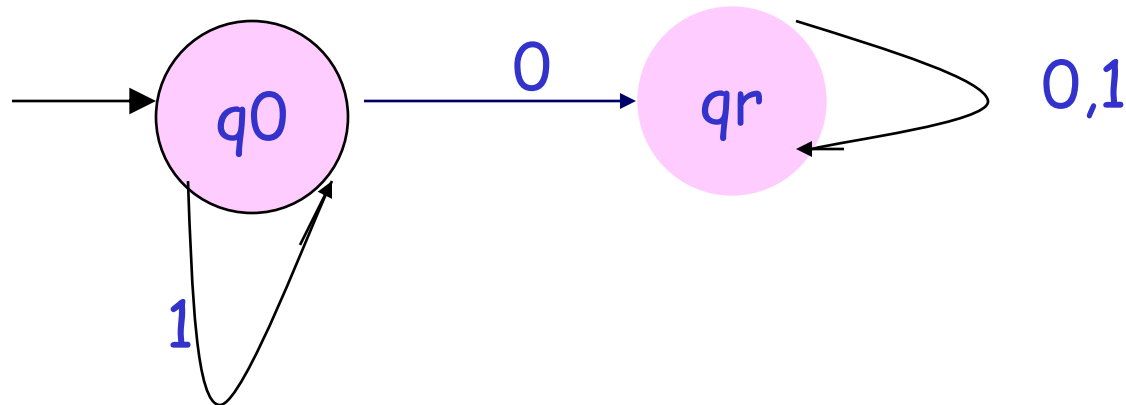
Exemplo

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



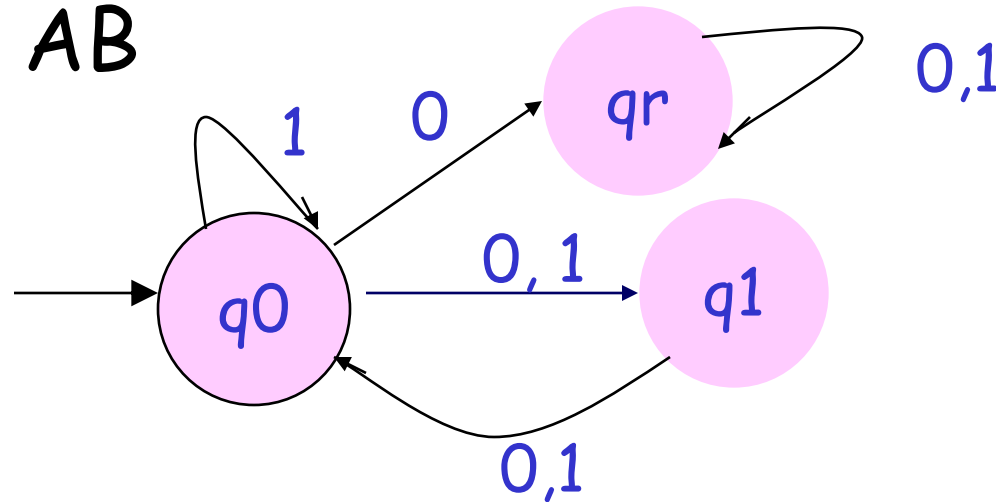
$$L(A) = \{(0+1)^{2n} \mid n=0,1,\dots\}$$

- 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.
- $L(AB) = \{0,1\}^* - \{0^{2n+1} \mid n \geq 0\}$

Operações que preservam a propriedade de ser
uma LR

Operações sobre LR que resultam em LR:

união

concatenação

fechamento

complemento

intersecção

diferença

reverso

Um AF não é um programa.....

....mas serve como um bom modelo. Veja no Material HTML de Referência - Tópicos em Teoria da Computação, em

<http://www.icmc.usp.br/~gracan/teoria/Teoria.html>

como podemos implementar um AFD:

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