Teoria da Computação

Autômatos Finitos e Linguagens Regulares

Aula 02

Prof. Felipe A. Louza

Roteiro

Roteiro

Autômatos Finitos

Vamos descrever autômatos finitos (AF) como *processadores de cadeias*.

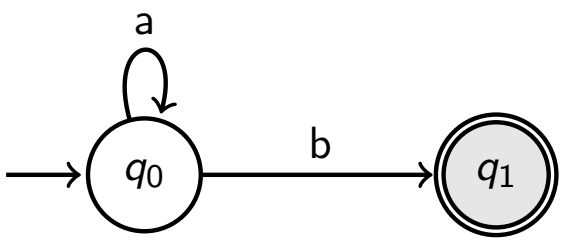


Figura: Diagrama de estados.

Componentes:

- Um estado q_0 é definido como inicial.
- Para cada símbolo, definimos uma função de transição para outros estados.
- Um conjunto de estados são designados como de aceitação ou finais.

Vamos representar um AF como um *grafo direcionado* (diagrama de estados).

Autômatos Finitos

Vamos descrever autômatos finitos (AF) como *processadores de cadeias*.

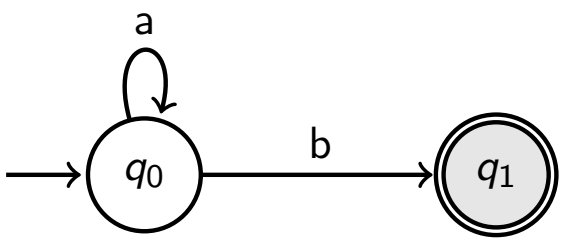


Figura: Diagrama de estados.

Componentes:

- Um estado q_0 é definido como inicial
- Para cada símbolo, definimos uma função de transição de saída para outros estados
- Um conjunto de estados são designados como de aceitação ou final

---

Vamos representar um AF como um *grafo direcionado* (diagrama de estados).

Autômatos Finitos

Vamos descrever autômatos finitos (AF) como *processadores de cadeias*.

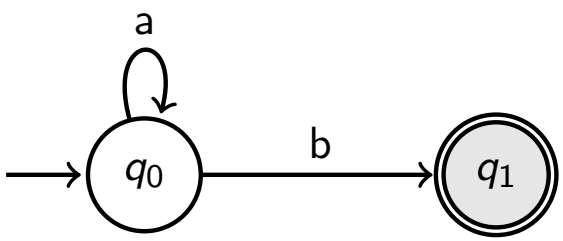


Figura: Diagrama de estados.

Componentes:

- Um estado q_0 é definido como inicial
- Para cada símbolo, definimos uma função de transição de saída para outros estados
- Um conjunto de estados são designados como de aceitação ou final

---

Vamos representar um AF como um *grafo direcionado* (diagrama de estados).

Autômatos Finitos

Vamos descrever autômatos finitos (AF) como *processadores de cadeias*.

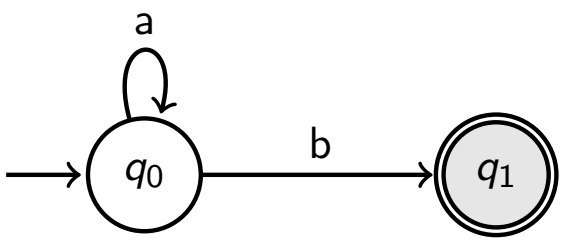


Figura: Diagrama de estados.

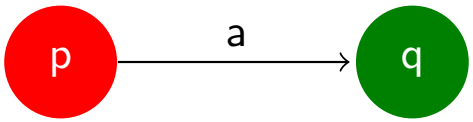
Componentes:

- Um estado q_0 é definido como inicial
- Para cada símbolo, definimos uma função de transição de saída para outros estados
- Um conjunto de estados são designados como de aceitação ou final

---

Vamos representar um AF como um *grafo direcionado* (diagrama de estados).

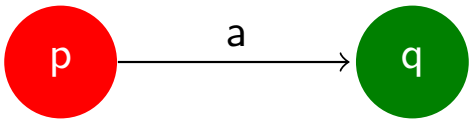
Autômatos Finitos



Detalhes:

- p e q são estados
- p estado atual
- a símbolo lido
- q estado novo
- Função de transição: $\delta(p, a) = q$

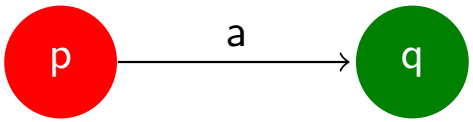
Autômatos Finitos



Detalhes:

- p e q são estados
- p estado atual
- a símbolo lido
- q estado novo
- Função de transição: $\delta(p, a) = q$

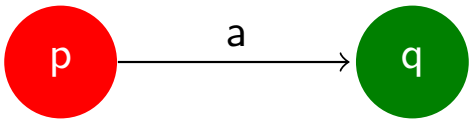
Autômatos Finitos



Detalhes:

- p e q são estados
- p estado atual
- a símbolo lido
- q estado novo
- Função de transição: $\delta(p, a) = q$

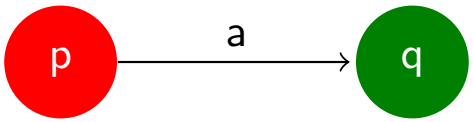
Autômatos Finitos



Detalhes:

- p e q são estados
- p estado atual
- a símbolo lido
- q estado novo
- Função de transição: $\delta(p, a) = q$

Autômatos Finitos



Detalhes:

- p e q são estados
- p estado atual
- a símbolo lido
- q estado novo
- Função de transição: $\delta(p, a) = q$

Autômatos Finitos

Autômatos finitos como *processadores de cadeias*:

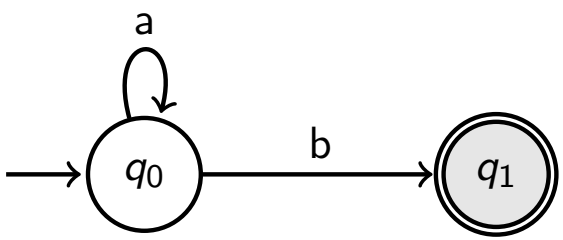


Figura: Diagrama de estados.

- A *saída* será **aceita** ou **rejeita** a cadeia w.
- O AF começa no estado inicial e $w = w_1 w_2 \ldots w_n$ é lida da esquerda para a direita.
- Após ler cada símbolo, o AF move-se de um estado para outro de acordo com função de transição.
- Uma cadeia w é **aceita** se ao processarmos $w_1 w_2 \ldots w_n$ o AF chega em um *estado de aceitação*.

Autômatos Finitos

Autômatos finitos como *processadores de cadeias*:

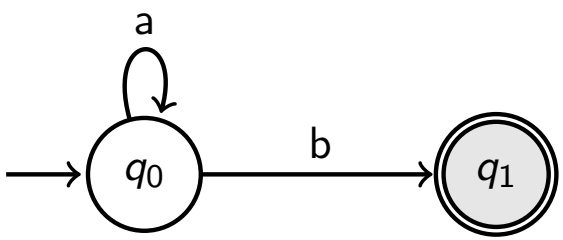


Figura: Diagrama de estados.

- A *saída* será **aceita** ou **rejeita** a cadeia w.
- O AF começa no estado inicial e $w = w_1 w_2 \ldots w_n$ é lida da esquerda para a direita.
- Após ler cada símbolo, o AF move-se de um estado para outro de acordo com função de transição.
- Uma cadeia w é **aceita** se ao processarmos $w_1 w_2 \ldots w_n$ o AF chega em um *estado de aceitação*.

Autômatos Finitos

Autômatos finitos como *processadores de cadeias*:

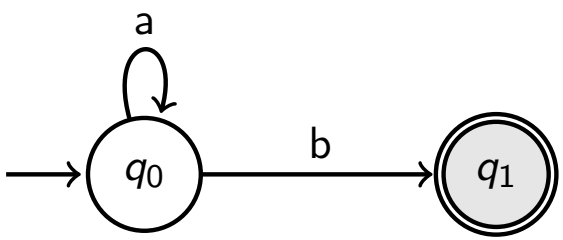


Figura: Diagrama de estados.

- A *saída* será **aceita** ou **rejeita** a cadeia w.
- O AF começa no estado inicial e $w = w_1 w_2 \ldots w_n$ é lida da esquerda para a direita.
- Após ler cada símbolo, o AF move-se de um estado para outro de acordo com função de transição.
- Uma cadeia w é **aceita** se ao processarmos $w_1 w_2 \ldots w_n$ o AF chega em um *estado de aceitação*.

Autômatos Finitos

Autômatos finitos como *processadores de cadeias*:

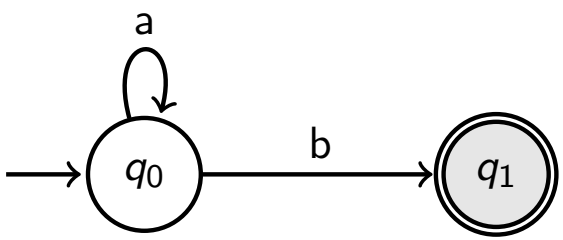


Figura: Diagrama de estados.

- A *saída* será **aceita** ou **rejeita** a cadeia w.
- O AF começa no estado inicial e $w = w_1 w_2 \ldots w_n$ é lida da esquerda para a direita.
- Após ler cada símbolo, o AF move-se de um estado para outro de acordo com função de transição.
- Uma cadeia w é **aceita** se ao processarmos $w_1 w_2 \ldots w_n$ o AF chega em um *estado de aceitação*.

Autômatos Finitos

Exemplo:

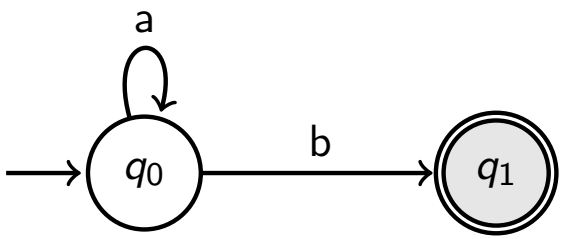


Figura: Diagrama de estados.

- As cadeias $w = ab, aab, aaab, aaaab, b, \ldots$ são aceitas.
- Em outras palavras, todas as cadeias $w = a^n b$, $n \geq 0$ são aceitas.

Exemplo:

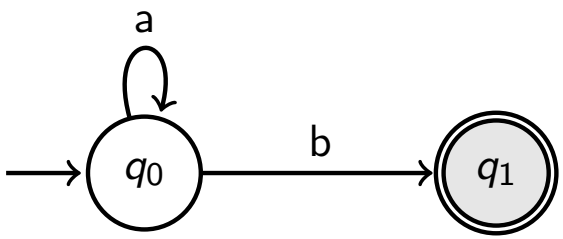
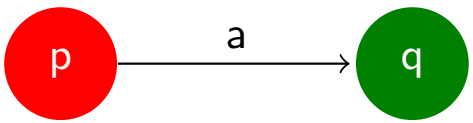


Figura: Diagrama de estados.

- As cadeias $w = ab, aab, aaab, aaaab, b, \ldots$ são aceitas.
- Em outras palavras, todas as cadeias $w = a^n b$, $n \geq 0$ são aceitas.

Autômatos Finitos

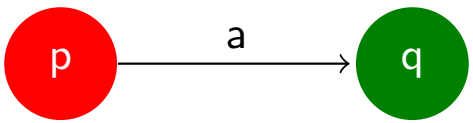


Começaremos com o formalismo de um *AF determinístico* (AFD):

- O termo *"determinístico"* se refere ao fato de que, existe somente um estado $\delta(p, a) = q$ ao qual o autômato pode transitar ao ler $w_i = a$.

AF não-determinísticos serão vistos na Seção 1.

Autômatos Finitos



Começaremos com o formalismo de um *AF determinístico* (AFD):

- O termo *"determinístico"* se refere ao fato de que, existe somente um estado $\delta(p, a) = q$ ao qual o autômato pode transitar ao ler $w_i = a$.

AF não-determinísticos serão vistos na Seção 1.

Roteiro

Definição formal de AFD

Definição

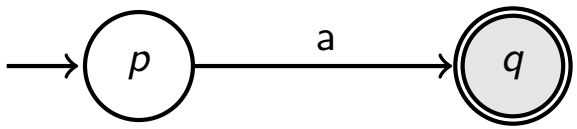
Um autômato finito determinístico (AFD) é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, em que:

1. Q é um conjunto finito de estados;
2. Σ é o alfabeto da cadeia de entrada;
3. $\delta : Q \times \Sigma \to Q$ é a função de transição;
   - $\delta(p, a) = q$
4. $q_0 \in Q$ é o estado inicial; e
5. $F \subseteq Q$ é o conjunto de estados de aceitação.

Definição formal de AFD

Função transição

- $\delta(p, a) = q$ significa que **quando** o AF está no estado p e **lê** o símbolo 'a', o próximo estado será q.



$$\delta(p, a) = q$$

	a	b
$\rightarrow p$	q	$\perp$
$\star q$	$\perp$	$\perp$

Tabela: Transições

δ pode não ser total: a função é indefinida ($\perp$) para alguns elementos de $Q \times \Sigma$

Definição formal de AFD

Condições de parada: após processar o último símbolo de $w_1 w_2 \ldots w_n$.

- **aceita** a entrada: ✓
  - se o AF assume um estado de aceitação.
- **rejeita** a entrada: ✗
  - se o AF não assume um estado de aceitação.
  - **OU** função não está definida para alguma transição $\delta(q_i, w_j) = \perp$.

O AFD sempre para: (não há loop infinito)

- Um novo símbolo é lido a cada aplicação da função de transição.

Definição formal de AFD

Condições de parada: após processar o último símbolo de $w_1 w_2 \ldots w_n$.

- **aceita** a entrada: ✓
  - se o AF assume um estado de aceitação.
- **rejeita** a entrada: ✗
  - se o AF não assume um estado de aceitação.
  - **OU** função não está definida para alguma transição $\delta(q_i, w_j) = \perp$.

O AFD sempre para: (não há loop infinito)

- Um novo símbolo é lido a cada aplicação da função de transição.

Exemplo de um autômato finito M_1

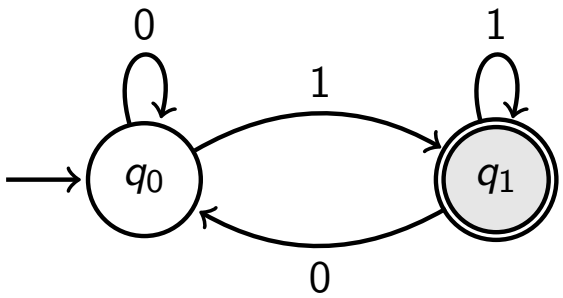
Seja $M_1 = (Q, \Sigma, \delta, q_0, F)$ um AF:

1. $Q = \{q_0, q_1\}$;
2. $\Sigma = \{0, 1\}$;
3. $\delta : Q \times \Sigma \to Q$ é descrita como:

	0	1
$\to q_0$	q_0	q_1
$\star q_1$	q_0	q_1

4. q_0 é o estado inicial; e
5. $F = \{q_1\}$.

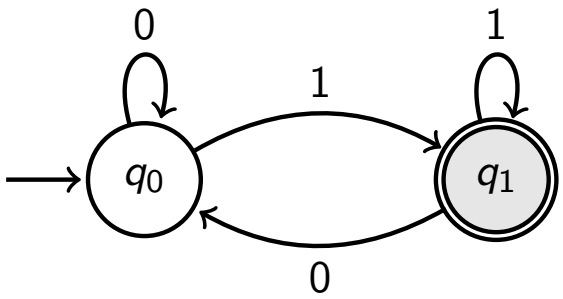
Exemplo de um autômato finito M_1



Características do diagrama de estado de M_1:

- 2 estados, $q_0, q_1 \in Q$
- Estado inicial $= q_0$
- Estado de aceitação $= q_1$
- Transições, arcos (arestas direcionadas) de um estado para outro

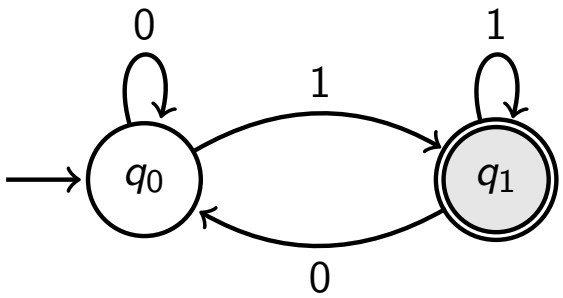
Exemplo de um autômato finito M_1



Características do diagrama de estado de M_1:

- 2 estados, $q_0, q_1 \in Q$
- Estado inicial $= q_0$
- Estado de aceitação $= q_1$
- Transições, arcos (arestas direcionadas) de um estado para outro

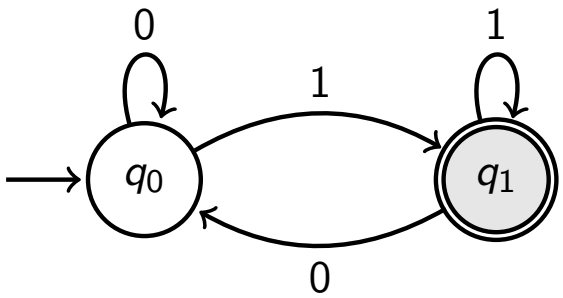
Autômato finito em ação



M_1 aceita ou rejeita entrada?
Entrada: $w = 1101$

- Inicia em q_0
- Le 1, transição de $q_0 \to q_1$
- Le 1, transição de $q_1 \to q_1$
- Le 0, transição de $q_1 \to q_0$
- Le 1, transição de $q_0 \to q_1$
- Aceita pois M_1 está no estado de aceitação $q_1 \in F$

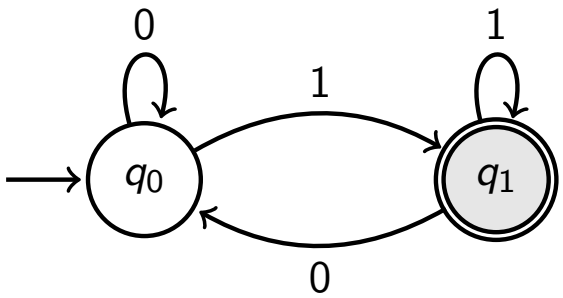
Autômato finito em ação



M_1 aceita ou rejeita entrada?
Entrada: $w = 1101$

- Inicia em q_0
- Le 1, transição de $q_0 \rightarrow q_1$
- Le 1, transição de $q_1 \rightarrow q_1$
- Le 0, transição de $q_1 \rightarrow q_0$
- Le 1, transição de $q_0 \rightarrow q_1$
- Aceita pois M_1 está no estado de aceitação $q_1 \in F$

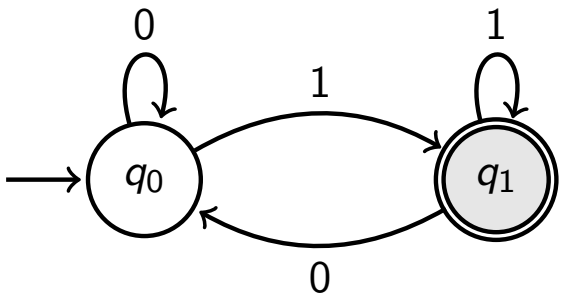
Autômato finito em ação



M_1 aceita ou rejeita entrada?
Entrada: $w = 1101$

- Inicia em q_0
- Le 1, transição de $q_0 \to q_1$
- Le 1, transição de $q_1 \to q_1$
- Le 0, transição de $q_1 \to q_0$
- Le 1, transição de $q_0 \to q_1$
- Aceita pois M_1 está no estado de aceitação $q_1 \in F$

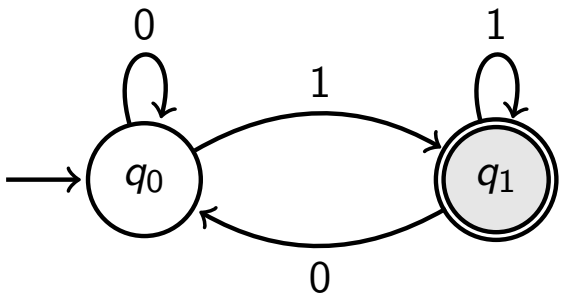
Autômato finito em ação



M_1 aceita ou rejeita entrada?
Entrada: $w = 1101$

- Inicia em q_0
- Le 1, transição de $q_0 \to q_1$
- Le 1, transição de $q_1 \to q_1$
- Le 0, transição de $q_1 \to q_0$
- Le 1, transição de $q_0 \to q_1$
- Aceita pois M_1 está no estado de aceitação $q_1 \in F$

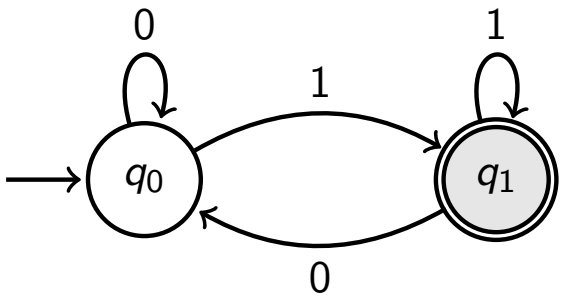
Autômato finito em ação



M_1 aceita ou rejeita entrada?
Entrada: $w = 1101$

- Inicia em q_0
- Le 1, transição de $q_0 \to q_1$
- Le 1, transição de $q_1 \to q_1$
- Le 0, transição de $q_1 \to q_0$
- Le 1, transição de $q_0 \to q_1$
- Aceita pois M_1 está no estado de aceitação $q_1 \in F$

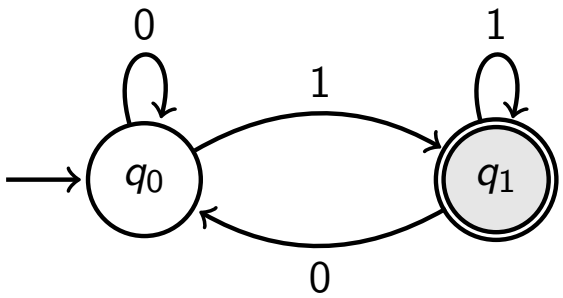
Autômato finito em ação



M_1 aceita ou rejeita entrada?
Entrada: $w = 1101$

- Inicia em q_0
- Le 1, transição de $q_0 \to q_1$
- Le 1, transição de $q_1 \to q_1$
- Le 0, transição de $q_1 \to q_0$
- Le 1, transição de $q_0 \to q_1$
- Aceita pois M_1 está no estado de aceitação $q_1 \in F$

Autômato finito e linguagens



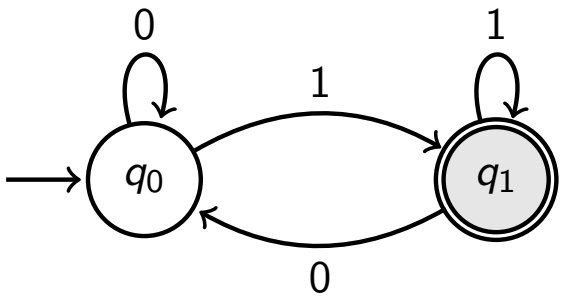
Linguagem *aceita* por M_1:

- Cadeias como:
  $01, 1, 0101, 11, 111, 1111, 1001, 100001111, 10000010101$
- Qualquer cadeia $w \in \{0,1\}^+$ que termine com 1.

Rejeita: $0, 10, 111110 \ldots$

Pergunta: Podemos descrever uma linguagem aceita por M_1?

Autômato finito e linguagens



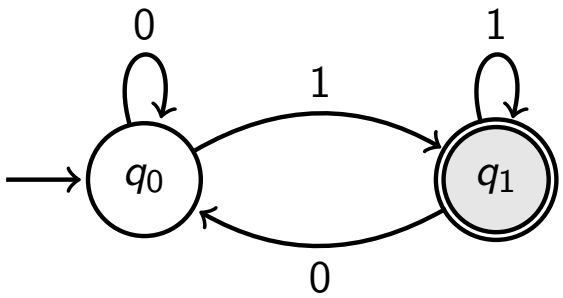
Linguagem *aceita* por M_1:

- Cadeias como:
  $01, 1, 0101, 11, 111, 1111, 1001, 100001111, 10000010101$
- Qualquer cadeia $w \in \{0,1\}^+$ que termine com 1.

Rejeita: $0, 10, 111110 \ldots$

Pergunta: Podemos descrever uma linguagem aceita por M_1?

Autômato finito e linguagens



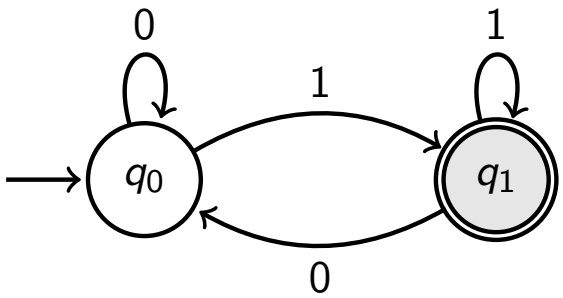
Linguagem *aceita* por M_1:

- Cadeias como:
  $01, 1, 0101, 11, 111, 1111, 1001, 100001111, 10000010101$
- Qualquer cadeia $w \in \{0,1\}^+$ que termine com 1.

Rejeita: $0, 10, 111110 \ldots$

Pergunta: Podemos descrever uma linguagem aceita por M_1?

Autômato finito e linguagens



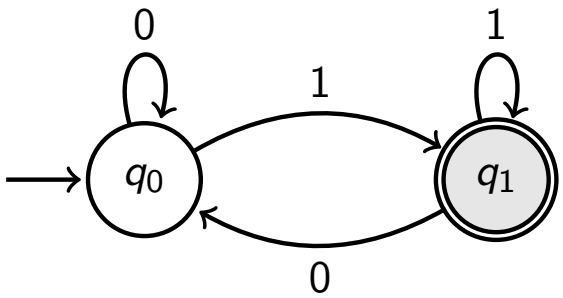
Linguagem *aceita* por M_1:

- Cadeias como:
  $01, 1, 0101, 11, 111, 1111, 1001, 100001111, 10000010101$
- Qualquer cadeia $w \in \{0,1\}^+$ que termine com 1.

Rejeita: $0, 10, 111110 \ldots$

Pergunta: Podemos descrever uma linguagem aceita por M_1?

Autômato finito e linguagens



Linguagem *aceita* por M_1:

- Cadeias como:
  $01, 1, 0101, 11, 111, 1111, 1001, 100001111, 10000010101$
- Qualquer cadeia $w \in \{0,1\}^+$ que termine com 1.

Rejeita: $0, 10, 111110 \ldots$

Pergunta: Podemos descrever uma linguagem aceita por M_1?

Importante!

Definição

Seja A o conjunto de todas as cadeias (palavras) aceitas por M, dizemos que A é a linguagem aceita por M, e escrevemos

$$L(M) = A$$

Dizemos que M reconhece ou aceita uma linguagem A se A linguagem A é finita.

Importante!

Definição

Seja A o conjunto de todas as cadeias (palavras) aceitas por M, dizemos que A é a linguagem aceita por M, e escrevemos

$$L(M) = A$$

- Dizemos que M reconhece ou aceita uma linguagem A.
- A linguagem A é única.
- Inclusive pode ser unicamente a linguagem vazia $\emptyset$.

Importante!

Definição

Seja A o conjunto de todas as cadeias (palavras) aceitas por M, dizemos que A é a linguagem aceita por M, e escrevemos

$$L(M) = A$$

- Dizemos que M reconhece ou aceita uma linguagem A.
- A linguagem A é única.
  - Inclusive pode ser unicamente a linguagem vazia $\varnothing$.

Importante!

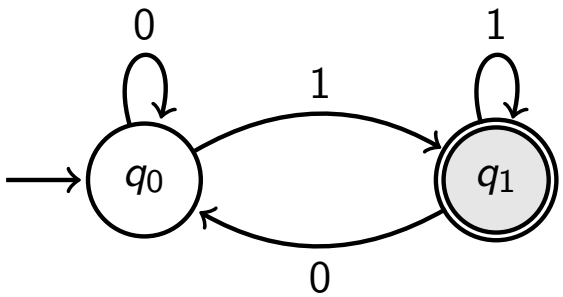
Definição

Seja A o conjunto de todas as cadeias (palavras) aceitas por M, dizemos que A é a linguagem aceita por M, e escrevemos

$$L(M) = A$$

- Dizemos que M reconhece ou aceita uma linguagem A.
- A linguagem A é única.
  - Inclusive pode ser unicamente a linguagem vazia $\varnothing$.

Linguagem de M_1

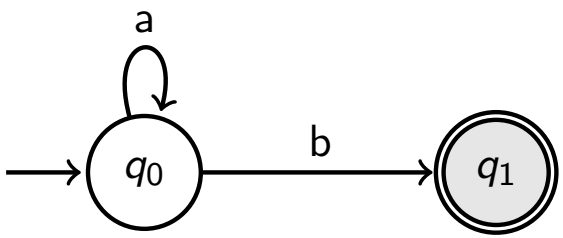


$$L(M_1) = \{w \mid w \text{ termina com } 1, \text{ e } \Sigma = \{0, 1\}\}$$

podemos dizer que M_1 reconhece $L(M_1)$

Autômato finito e linguagens

Outros exemplos:

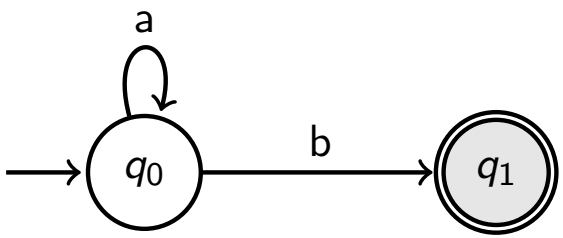


Linguagem *aceita* por M_2:

$$L(M_2) = \{a^n b \mid n \geq 0 \text{ e } \Sigma = \{a, b\}\}$$

Autômato finito e linguagens

Outros exemplos:

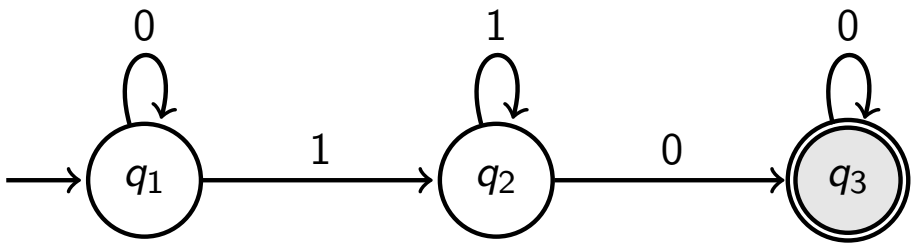


Linguagem *aceita* por M_2:

$$L(M_2) = \{a^n b \mid n \geq 0 \text{ e } \Sigma = \{a, b\}\}$$

Autômato finito e linguagens

Outros exemplos:

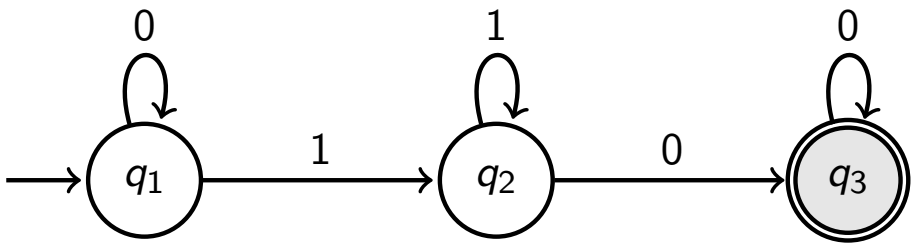


Linguagem *aceita* por M_3:

$$L(M_3) = \{w \mid w \text{ contém um número qualquer de 0s, pelo menos um 1 e pelo menos um 0}\}$$

Autômato finito e linguagens

Outros exemplos:

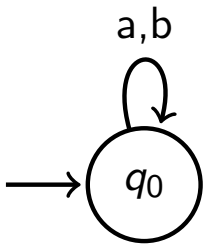


Linguagem *aceita* por M_3:

$$L(M_3) = \{w \mid w \text{ contém um número qualquer de 0s, pelo menos um 1 e pelo menos um 0}\}$$

Autômato finito e linguagens

Outros exemplos:

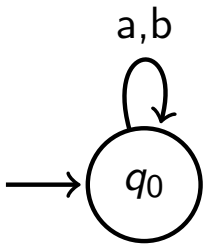


Linguagem *aceita* por M_4:

$L(M_4) = \varnothing$, ou seja, a linguagem vazia.

Autômato finito e linguagens

Outros exemplos:

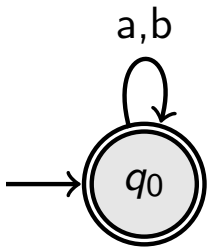


Linguagem *aceita* por M_4:

$$L(M_4) = \varnothing, \text{ ou seja, a linguagem vazia.}$$

Autômato finito e linguagens

Outros exemplos:



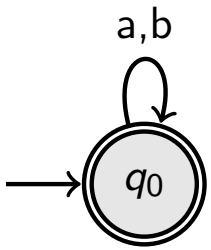
Linguagem *aceita* por M_5:

$$L(M_5) = \Sigma^*$$

Lembrando que $\mathcal{E} \in \Sigma^*$.

Autômato finito e linguagens

Outros exemplos:



Linguagem *aceita* por M_5:

$$L(M_5) = \Sigma^*$$

Lembrando que $\mathcal{E} \in \Sigma^*$.

Função Programa

Para descrever formalmente o comportamento de um AF, vamos estender a função de transição $\delta(p, a)$

Definição

A função de transição estendida (ou programa) de M, denotada por

$$\delta^* : Q \times \Sigma^* \to Q$$

é a função $\delta : Q \times \Sigma$ estendida para palavras, definida como:

$$\delta^*(q, \mathcal{E}) = q$$
$$\delta^*(q, aw) = \delta^*(\delta(q, a), w)$$

ou seja, δ^* é a sucessiva aplicação da função de transição para cada símbolo da palavra.

Função Programa

Para descrever formalmente o comportamento de um AF, vamos estender a função de transição $\delta(p, a)$

Definição

A função de transição estendida (ou programa) de M, denotada por

$$\delta^* : Q \times \Sigma^* \to Q$$

é a função $\delta : Q \times \Sigma$ estendida para palavras, definida como:

$$\delta^*(q, \mathcal{E}) = q$$
$$\delta^*(q, aw) = \delta^*(\delta(q, a), w)$$

ou seja, δ^* é a sucessiva aplicação da função de transição para cada símbolo da palavra.

Função Programa

Notação $\delta = \delta^*$

Para simplificar a notação, muitas vezes denotaremos ambas, a função δ e a função de transição estendida δ^*, com o mesmo símbolo δ.

$$\delta(q, w) = p, \text{quando } w \in \Sigma^*, \ \delta = \delta^*$$

Função Programa

Definição

Uma cadeia w é aceita pelo AF $M = (Q, \Sigma, \delta, q_0, F)$ se

$$\delta^*(q_0, w) = p, \text{ tal que } p \in F$$

Em outras palavras, M aceita $w = w_1 w_2 \ldots w_n$ se existe uma sequência de estados $r_0, r_1, \ldots, r_n$, tal que:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, para $i = 0, \ldots, n-1$, e
3. $r_n \in F$.

Função Programa

Definição

Uma cadeia w é aceita pelo AF $M = (Q, \Sigma, \delta, q_0, F)$ se

$$\delta^*(q_0, w) = p, \text{ tal que } p \in F$$

Em outras palavras, M aceita $w = w_1 w_2 \ldots w_n$ se existe uma sequência de estados $r_0, r_1, \ldots, r_n$, tal que:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, para $i = 0, \ldots, n-1$, e
3. $r_n \in F$.

Função Programa

Definição

Uma cadeia w é aceita pelo AF $M = (Q, \Sigma, \delta, q_0, F)$ se

$$\delta^*(q_0, w) = p, \text{ tal que } p \in F$$

Em outras palavras, M aceita $w = w_1 w_2 \ldots w_n$ se existe uma sequência de estados $r_0, r_1, \ldots, r_n$, tal que:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, para $i = 0, \ldots, n-1$, e
3. $r_n \in F$.

Função Programa

Definição

Uma cadeia w é aceita pelo AF $M = (Q, \Sigma, \delta, q_0, F)$ se

$$\delta^*(q_0, w) = p, \text{ tal que } p \in F$$

Em outras palavras, M aceita $w = w_1 w_2 \ldots w_n$ se existe uma sequência de estados $r_0, r_1, \ldots, r_n$, tal que:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, para $i = 0, \ldots, n-1$, e
3. $r_n \in F$.

Função Programa

Definição

Uma cadeia w é aceita pelo AF $M = (Q, \Sigma, \delta, q_0, F)$ se

$$\delta^*(q_0, w) = p, \text{ tal que } p \in F$$

Em outras palavras, M aceita $w = w_1 w_2 \ldots w_n$ se existe uma sequência de estados $r_0, r_1, \ldots, r_n$, tal que:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, para $i = 0, \ldots, n-1$, e
3. $r_n \in F$.

Função Programa

Explicando...

1. A máquina começa no estado inicial
2. A máquina avança para os próximos estados de acordo com a função de transição
3. A máquina aceita a entrada se termina num estado de aceitação

Aceitação:

- Dizemos que M reconhece a linguagem A se

$$A = \{w \mid M \text{ aceita } w\}$$

Função Programa

Explicando...

1. A máquina começa no estado inicial
2. A máquina avança para os próximos estados de acordo com a função de transição
3. A máquina aceita a entrada se termina num estado de aceitação

Aceitação:

- Dizemos que M reconhece a linguagem A se

$$A = \{w \mid M \text{ aceita } w\}$$

Exemplo

Considere o AFD $M_6 = (\{q_0, q_1, q_2, q_f\}, \{a, b\}, \delta, q_0, \{q_f\})$ com δ_1:

δ	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_f	q_2
q_2	q_1	q_f
$\star q_f$	q_f	q_f

Tabela: Função δ_1 (função programa) do AFD M_6.

Calcule o resultado da computação da palavra $w = abaa$ a partir de q_0.

Exemplo: continuação

Aplicação da palavra $w = abaa$ a partir de q_0.

$\delta^*(q_0, \text{abaa}) =$ função estendida sobre abaa

$\delta^*(\delta(q_0, a), \text{baa}) =$ processa abaa

$\delta^*(q_1, \text{baa}) =$ função estendida sobre baa

$\delta^*(\delta(q_1, b), \text{aa}) =$ processa baa

$\delta^*(q_2, \text{aa}) =$ função estendida sobre aa

$\delta^*(\delta(q_2, a), \text{a}) =$ processa aa

$\delta^*(q_1, \text{a}) =$ função estendida sobre a

$\delta^*(\delta(q_1, a), \mathcal{E}) =$ processa a

$\delta^*(q_f, \mathcal{E}) = q_f$ função estendida sobre $\mathcal{E}$

e, deste modo, a palavra $w = abaa$ é aceita pois o AFD para no estado final.

Exemplo: continuação

Aplicação da palavra $w = abaa$ a partir de q_0.

$\delta^*(q_0, \text{abaa}) =$ função estendida sobre abaa
$\delta^*(\delta(q_0, a), \text{baa}) =$ processa abaa
$\delta^*(q_1, \text{baa}) =$ função estendida sobre baa
$\delta^*(\delta(q_1, b), \text{aa}) =$ processa baa
$\delta^*(q_2, \text{aa}) =$ função estendida sobre aa
$\delta^*(\delta(q_2, a), \text{a}) =$ processa aa
$\delta^*(q_1, \text{a}) =$ função estendida sobre a
$\delta^*(\delta(q_1, a), \mathcal{E}) =$ processa a
$\delta^*(q_f, \mathcal{E}) = q_f$ função estendida sobre $\mathcal{E}$

e, deste modo, a palavra $w = abaa$ é aceita pois o AFD para no estado final.

Exemplo: continuação

Aplicação da palavra $w = abaa$ a partir de q_0.

$\delta^*(q_0, \text{abaa}) =$ função estendida sobre abaa
$\delta^*(\delta(q_0, a), \text{baa}) =$ processa abaa
$\delta^*(q_1, \text{baa}) =$ função estendida sobre baa
$\delta^*(\delta(q_1, b), \text{aa}) =$ processa baa
$\delta^*(q_2, \text{aa}) =$ função estendida sobre aa
$\delta^*(\delta(q_2, a), \text{a}) =$ processa aa
$\delta^*(q_1, \text{a}) =$ função estendida sobre a
$\delta^*(\delta(q_1, a), \mathcal{E}) =$ processa a
$\delta^*(q_f, \mathcal{E}) = q_f$ função estendida sobre $\mathcal{E}$

e, deste modo, a palavra $w = abaa$ é aceita pois o AFD para no estado final.

Exemplo: continuação

Aplicação da palavra $w = abaa$ a partir de q_0.

$\delta^*(q_0, \text{abaa}) =$ função estendida sobre abaa
$\delta^*(\delta(q_0, a), \text{baa}) =$ processa abaa
$\delta^*(q_1, \text{baa}) =$ função estendida sobre baa
$\delta^*(\delta(q_1, b), \text{aa}) =$ processa baa
$\delta^*(q_2, \text{aa}) =$ função estendida sobre aa
$\delta^*(\delta(q_2, a), \text{a}) =$ processa aa
$\delta^*(q_1, \text{a}) =$ função estendida sobre a
$\delta^*(\delta(q_1, a), \mathcal{E}) =$ processa a
$\delta^*(q_f, \mathcal{E}) = q_f$ função estendida sobre $\mathcal{E}$

e, deste modo, a palavra $w = abaa$ é aceita pois o AFD para no estado final.

Exemplo: continuação

Aplicação da palavra $w = abaa$ a partir de q_0.

$\delta^*(q_0, \text{abaa}) =$	função estendida sobre abaa
$\delta^*(\delta(q_0, a), \text{baa}) =$	processa abaa
$\delta^*(q_1, \text{baa}) =$	função estendida sobre baa
$\delta^*(\delta(q_1, b), \text{aa}) =$	processa baa
$\delta^*(q_2, \text{aa}) =$	função estendida sobre aa
$\delta^*(\delta(q_2, a), \text{a}) =$	processa aa
$\delta^*(q_1, \text{a}) =$	função estendida sobre a
$\delta^*(\delta(q_1, a), \mathcal{E}) =$	processa a
$\delta^*(q_f, \mathcal{E}) = q_f$	função estendida sobre $\mathcal{E}$

e, deste modo, a palavra $w = abaa$ é aceita pois o AFD para no estado final.

Exemplo: continuação

Aplicação da palavra $w = abaa$ a partir de q_0.

$\delta^*(q_0, \text{abaa}) =$	função estendida sobre abaa
$\delta^*(\delta(q_0, a), \text{baa}) =$	processa abaa
$\delta^*(q_1, \text{baa}) =$	função estendida sobre baa
$\delta^*(\delta(q_1, b), \text{aa}) =$	processa baa
$\delta^*(q_2, \text{aa}) =$	função estendida sobre aa
$\delta^*(\delta(q_2, a), \text{a}) =$	processa aa
$\delta^*(q_1, \text{a}) =$	função estendida sobre a
$\delta^*(\delta(q_1, a), \mathcal{E}) =$	processa a
$\delta^*(q_f, \mathcal{E}) = q_f$	função estendida sobre $\mathcal{E}$

e, deste modo, a palavra $w = abaa$ é aceita pois o AFD para no estado final.

Exemplo: continuação

Aplicação da palavra $w = abaa$ a partir de q_0.

$\delta^*(q_0, \text{abaa}) =$	função estendida sobre abaa
$\delta^*(\delta(q_0, a), \text{baa}) =$	processa abaa
$\delta^*(q_1, \text{baa}) =$	função estendida sobre baa
$\delta^*(\delta(q_1, b), \text{aa}) =$	processa baa
$\delta^*(q_2, \text{aa}) =$	função estendida sobre aa
$\delta^*(\delta(q_2, a), \text{a}) =$	processa aa
$\delta^*(q_1, \text{a}) =$	função estendida sobre a
$\delta^*(\delta(q_1, a), \mathcal{E}) =$	processa a
$\delta^*(q_f, \mathcal{E}) = q_f$	função estendida sobre $\mathcal{E}$

e, deste modo, a palavra $w = abaa$ é aceita pois o AFD para no estado final.

Exemplo: continuação

Aplicação da palavra $w = abaa$ a partir de q_0.

$\delta^*(q_0, \text{abaa}) =$	função estendida sobre abaa
$\delta^*(\delta(q_0, a), \text{baa}) =$	processa abaa
$\delta^*(q_1, \text{baa}) =$	função estendida sobre baa
$\delta^*(\delta(q_1, b), \text{aa}) =$	processa baa
$\delta^*(q_2, \text{aa}) =$	função estendida sobre aa
$\delta^*(\delta(q_2, a), \text{a}) =$	processa aa
$\delta^*(q_1, \text{a}) =$	função estendida sobre a
$\delta^*(\delta(q_1, a), \mathcal{E}) =$	processa a
$\delta^*(q_f, \mathcal{E}) = q_f$	função estendida sobre $\mathcal{E}$

e, deste modo, a palavra $w = abaa$ é aceita pois o AFD para no estado final.

Exemplo: continuação

Aplicação da palavra $w = abaa$ a partir de q_0.

$$
\begin{aligned}
\delta^*(q_0, \text{abaa}) &= && \text{função estendida sobre abaa} \\
\delta^*(\delta(q_0, a), \text{baa}) &= && \text{processa abaa} \\
\delta^*(q_1, \text{baa}) &= && \text{função estendida sobre baa} \\
\delta^*(\delta(q_1, b), \text{aa}) &= && \text{processa baa} \\
\delta^*(q_2, \text{aa}) &= && \text{função estendida sobre aa} \\
\delta^*(\delta(q_2, a), \text{a}) &= && \text{processa aa} \\
\delta^*(q_1, \text{a}) &= && \text{função estendida sobre a} \\
\delta^*(\delta(q_1, a), \mathcal{E}) &= && \text{processa a} \\
\delta^*(q_f, \mathcal{E}) &= q_f && \text{função estendida sobre } \mathcal{E}
\end{aligned}
$$

e, deste modo, a palavra $w = abaa$ é aceita pois o AFD para no estado final.

Função Programa

Atenção:

A função programa $\delta^* : Q \times \Sigma^* \to Q$ é uma função parcial que a partir do estado corrente e do símbolo lido, determina o novo estado do autômato (pode ser indefinida):

$$\delta(r_i, w_{i+1}) = \begin{cases} r_{i+1} \\ \perp \qquad \text{(indefinido)} \end{cases}$$

Ou seja, nem todas as possibilidades em $\delta^* : Q \times \Sigma^* \to Q$ são necessariamente definidas.

Uma função parcial $f \subseteq A \times B$, não força f mapear todos os elementos do domínio A no contradomínio B, só o subconjunto A' de A.

Função Programa

Atenção:

A função programa $\delta^* : Q \times \Sigma^* \rightarrow Q$ é uma função parcial que a partir do estado corrente e do símbolo lido, determina o novo estado do autômato (pode ser indefinida):

$$\delta(r_i, w_{i+1}) = \begin{cases} r_{i+1} \\ \perp \qquad \text{(indefinido)} \end{cases}$$

Ou seja, nem todas as possibilidades em $\delta^* : Q \times \Sigma^* \rightarrow Q$ são necessariamente definidas.

Uma função parcial $f \subseteq A \times B$, não força f mapear todos os elementos do domínio A no contradomínio B, só o subconjunto A' de A.

Linguagem aceita por um AF

Definição

A linguagem reconhecida pelo AF M, denotada por $L(M)$, é o conjunto de todas as palavras pertencentes a Σ^* aceitas por M a partir de q_0.
Ou seja,

$$L(M) = \{w \mid \delta^*(q_0, w) \in F\}$$

Analogamente, $REJEITA(M)$ é o conjunto de todas as palavras em Σ^* não aceitas por M.

- $REJEITA(M) = \{w \mid \delta^*(q_0, w) \notin F$ ou $\delta^*(q_0, w)$ é indefinida$\}$

Linguagem aceita por um AF

Definição

A linguagem reconhecida pelo AF M, denotada por $L(M)$, é o conjunto de todas as palavras pertencentes a Σ^* aceitas por M a partir de q_0. Ou seja,

$$L(M) = \{w \mid \delta^*(q_0, w) \in F\}$$

Analogamente, $REJEITA(M)$ é o conjunto de todas as palavras em Σ^* não aceitas por M.

- $REJEITA(M) = \{w \mid \delta^*(q_0, w) \notin F$ ou $\delta^*(q_0, w)$ é indefinida$\}$

Algumas relações:

- Dado um AF definido como: $M = (Q, \Sigma, \delta, q_0, F)$
- $L(M) \cup \text{REJEITA}(M) = \Sigma^*$
- $L(M) \cap \text{REJEITA}(M) = \varnothing$
- $\sim L(M) = \text{REJEITA}(M)$
- $\sim \text{REJEITA}(M) = L(M)$

Roteiro

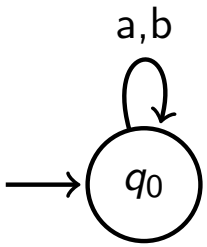
Linguagem Regular

Definição

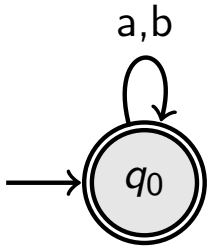
Uma linguagem L é dita uma Linguagem Regular se existe pelo menos um AF que aceita L.

Exemplos

Então, temos que $L_4 = \varnothing$ e $L_5 = \Sigma^*$ são Linguagens Regulares.



$L(M_4) = \varnothing$



$L(M_5) = \Sigma^*$

Definição: Linguagem Regular

Observações:

- Diferentes AFs podem aceitar uma mesma linguagem.
- M_1 e M_2 são Autômatos Finitos *Equivalentes* se e somente se:

$$L(M_1) = L(M_2)$$

Definição: Linguagem Regular

Observações:

- Diferentes AFs podem aceitar uma mesma linguagem.
- M_1 e M_2 são Autômatos Finitos *Equivalentes* se e somente se:

$$L(M_1) = L(M_2)$$

Hierarquia de Chomsky

Tipos de linguagens:

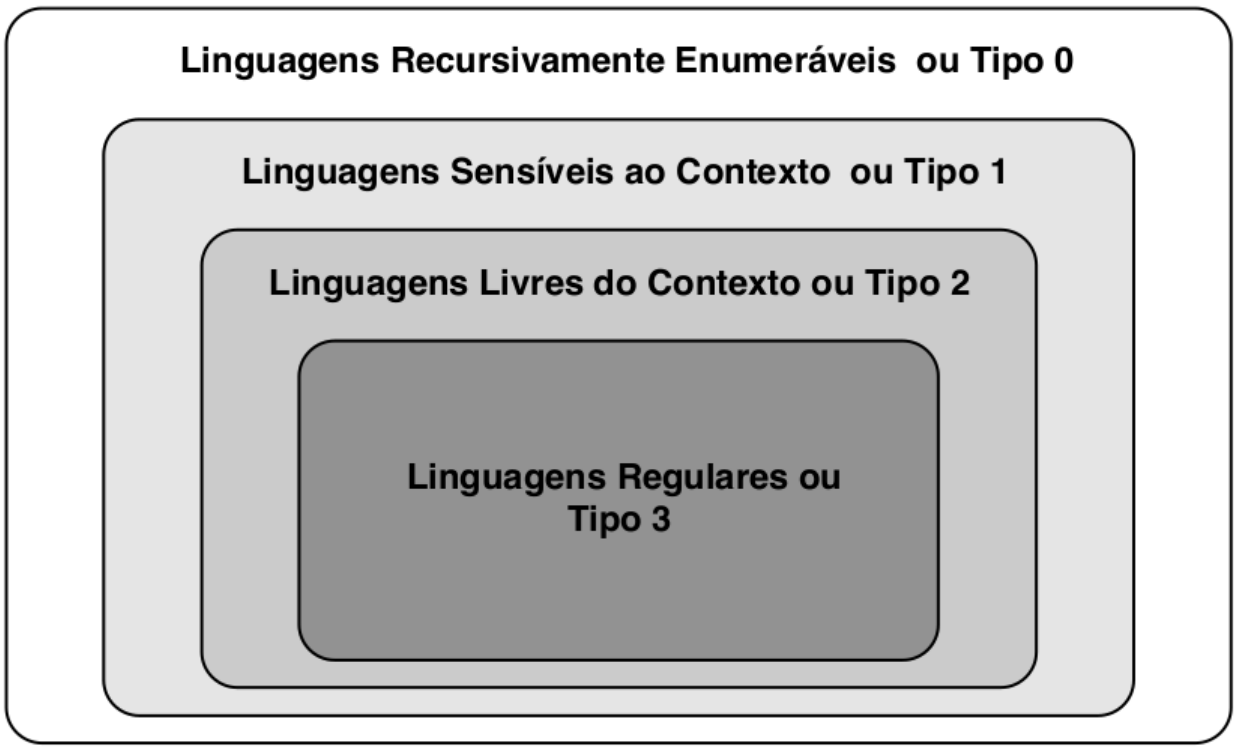
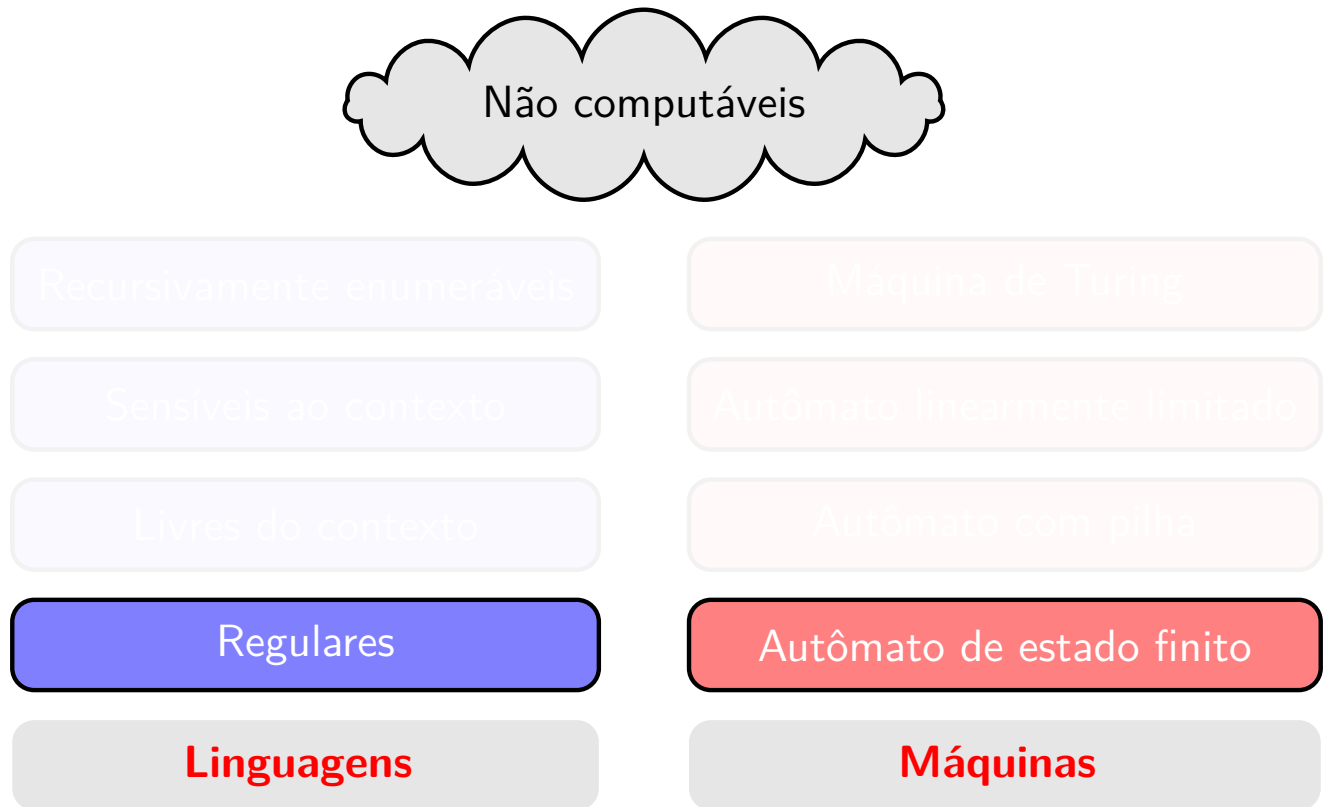


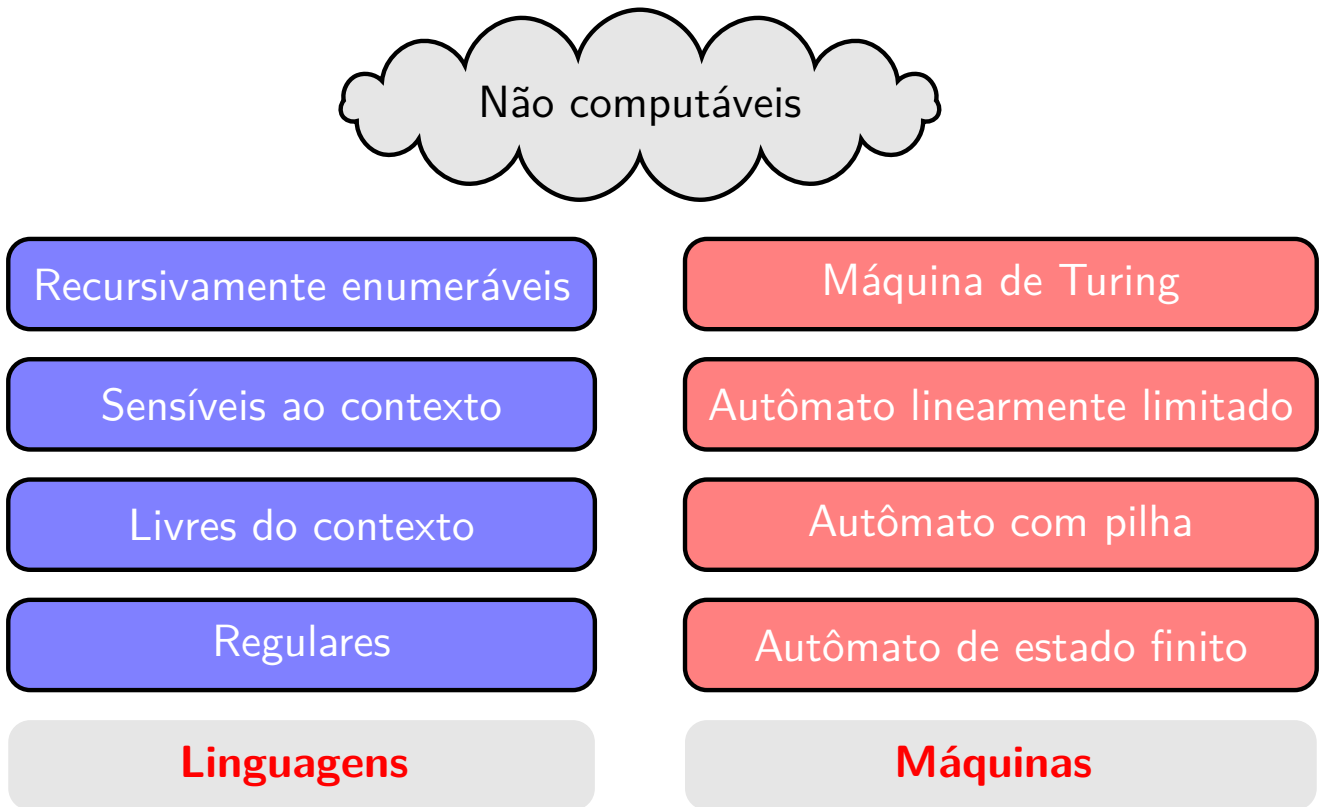
Figura: Hierarquia de Chomsky

Chomsky definiu estas classes como (potenciais) modelos para ling. naturais.

Hierarquia de Chomsky



Hierarquia de Chomsky



Roteiro

1. Autômatos Finitos
   - Autômato Finito Determinístico
   - Formalização de um AFD
   - Linguagens Regulares

2. Autômatos finitos não-determinísticos (AFNs)
   - Não-determinismo
   - Formalização de um AFN
   - Equivalência entre AFD e AFN

3. Referências

Roteiro

Autômatos finitos não-determinísticos

Vamos introduzir o conceito de Autômato finito *não-determinístico* (AFN):

- Vamos considerar um AF que permite zero, uma ou mais transições de um estado p, com o mesmo símbolo $a \in \Sigma$:

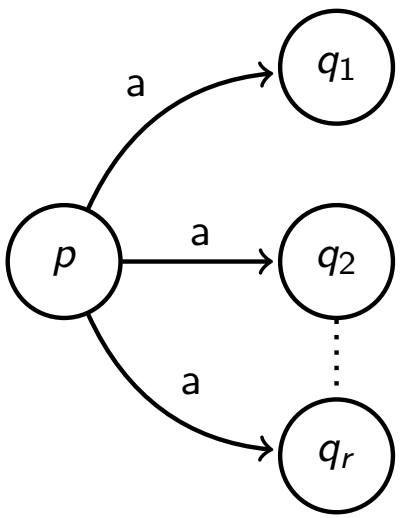
$$\delta(p, a) = \{q_1, q_2, \ldots q_r\}$$

Autômatos finitos não-determinísticos

Várias escolhas podem existir para o próximo estado em qualquer ponto.

Função de transição (mapeamento)

- $\delta(p, a) = \{q_1, q_2, \ldots q_r\}$



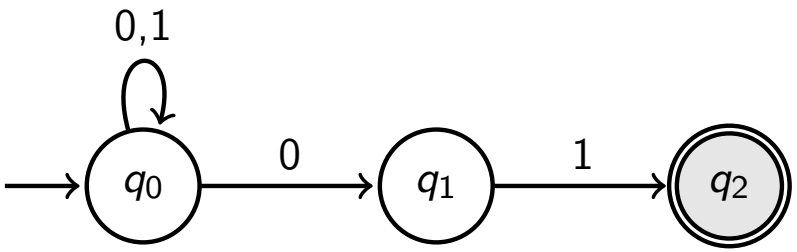
AFN

	a
p	$\{q_1, q_2, \ldots, q_r\}$

Tabela de transições

Autômatos finitos não-determinísticos

Exemplo:

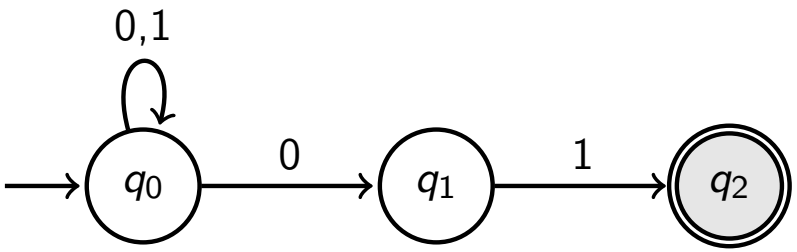


Linguagem *aceita* por N_7:

$$L = \{w | w \in \{0,1\}^* \text{ e w termina com } 01\}$$

Autômatos finitos não-determinísticos

Exemplo:



Linguagem *aceita* por N_7:

$$L = \{w | w \in \{0,1\}^* \text{ e w termina com } 01\}$$

Autômatos finitos não-determinísticos

Uma cadeia $w = w_1 w_2 \ldots w_n$ é **aceita** por um AFN se existir pelo menos uma sequência de transições que leva $\delta^*(q_0, w)$ a um estado de aceitação ao processar w.

- Podemos *imaginar* que a máquina se divide em *múltiplas cópias* de si mesma e segue todas as possibilidades em paralelo.

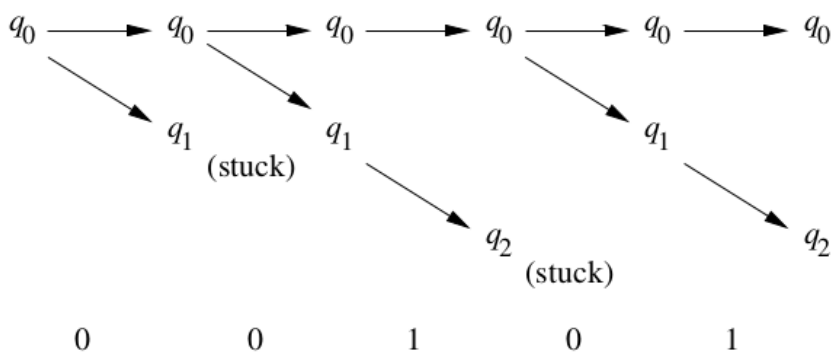


Figura: Considere N_1 e $w = 00101$

Autômatos finitos não-determinísticos

AFNs são uma generalização de AFDs:

- Todo AFD é um AFN.
- Em um AFN podem existir *vários caminhos* para a aceitação de w, enquanto que em um AFD, apenas um $\delta^*(q_0, w) \in F$.

Autômatos finitos não-determinísticos

AFNs são uma generalização de AFDs:

- Todo AFD é um AFN.
- Em um AFN podem existir *vários caminhos* para a aceitação de w, enquanto que em um AFD, apenas um $\delta^*(q_0, w) \in F$.

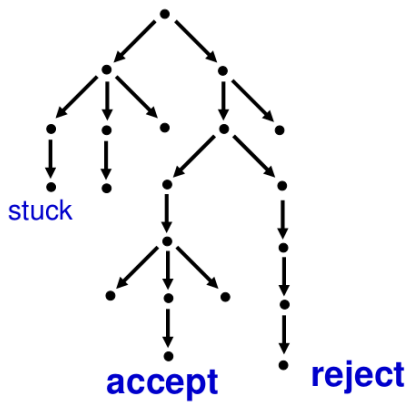


Figura: AFN

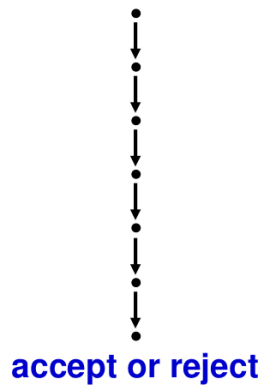
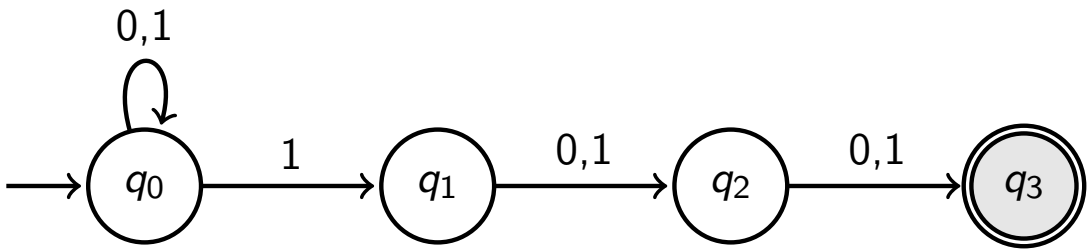


Figura: AFD

Autômatos finitos não-determinísticos

Outro exemplo:



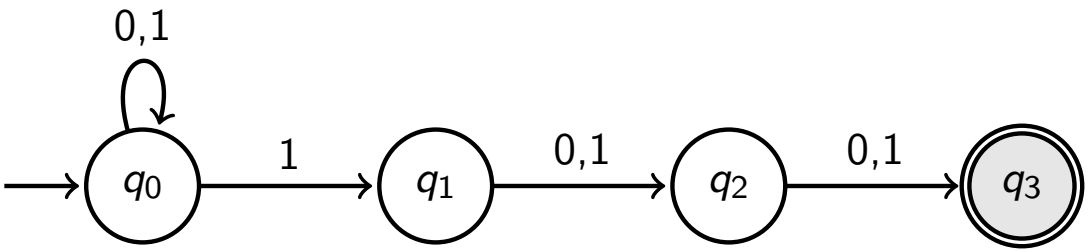
Linguagem *aceita* por N_8:

$$L = \{w \mid w \in \{0,1\}^* \text{ e } w \text{ possui um 1 em } w_{n-2}\}$$

Uma outra maneira de ver a computação nesse AFN é a de que ele permanece em q_0 até que "adivinhe" que está no final de w

Autômatos finitos não-determinísticos

Outro exemplo:



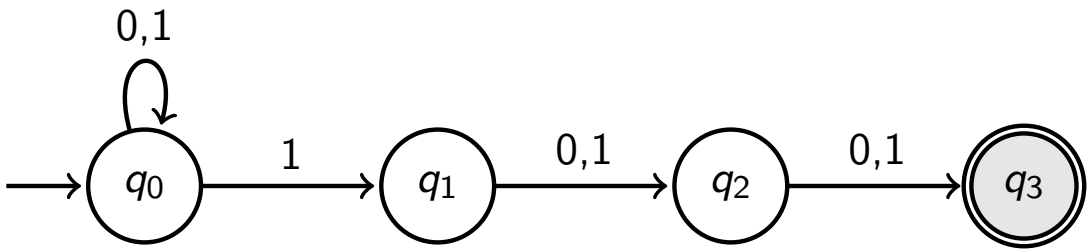
Linguagem *aceita* por N_8:

$$L = \{w \mid w \in \{0,1\}^* \text{ e } w \text{ possui um 1 em } w_{n-2}\}$$

- Uma outra maneira de ver a computação nesse AFN é dizer que ele permanece em q_0 até que *"adivinhe"* que está no final de w.
  - Nesse ponto, ele ramifica para o estado q_1, e segue para q_2 e q_3.

Autômatos finitos não-determinísticos

Outro exemplo:



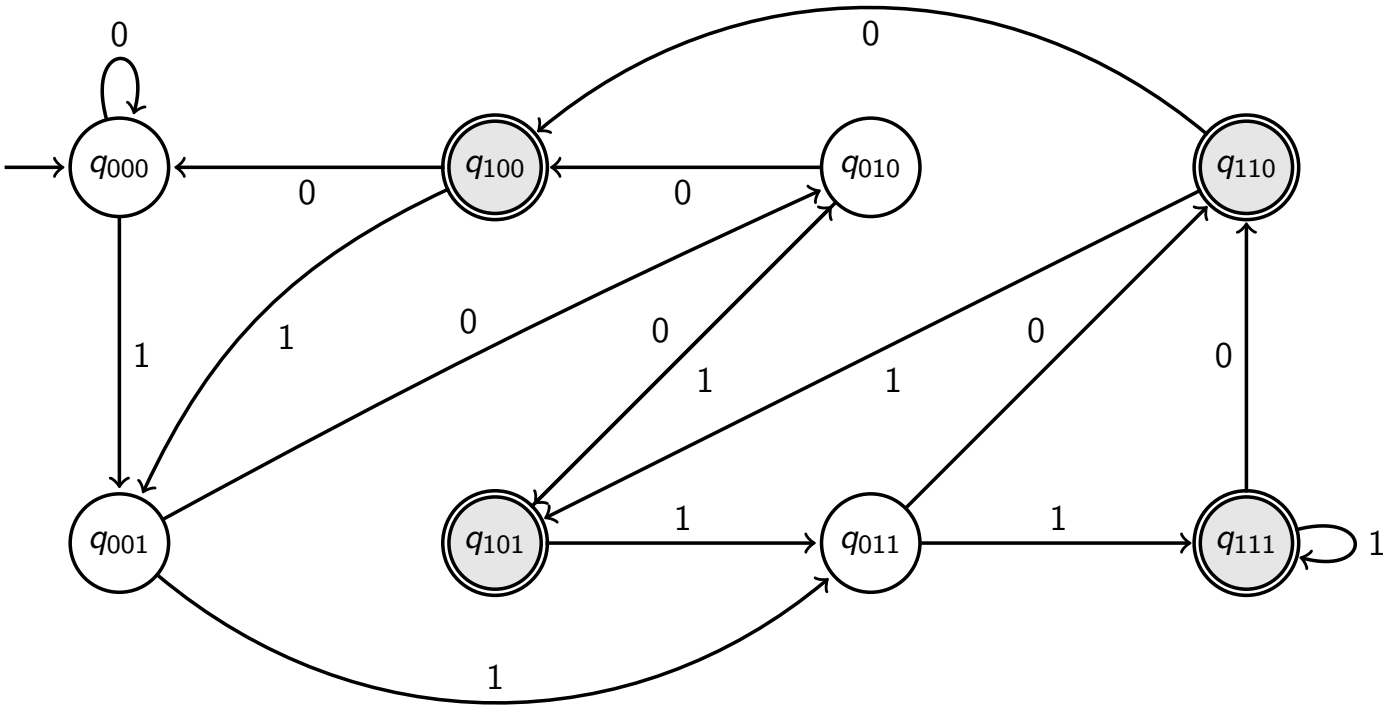
Linguagem *aceita* por N_8:

$$L = \{w \mid w \in \{0,1\}^* \text{ e } w \text{ possui um 1 em } w_{n-2}\}$$

- Uma outra maneira de ver a computação nesse AFN é dizer que ele permanece em q_0 até que *"adivinhe"* que está no final de w.
  - Nesse ponto, ele ramifica para o estado q_1, e segue para q_2 e q_3.

Autômatos finitos não-determinísticos

AFD equivalente à N_8:

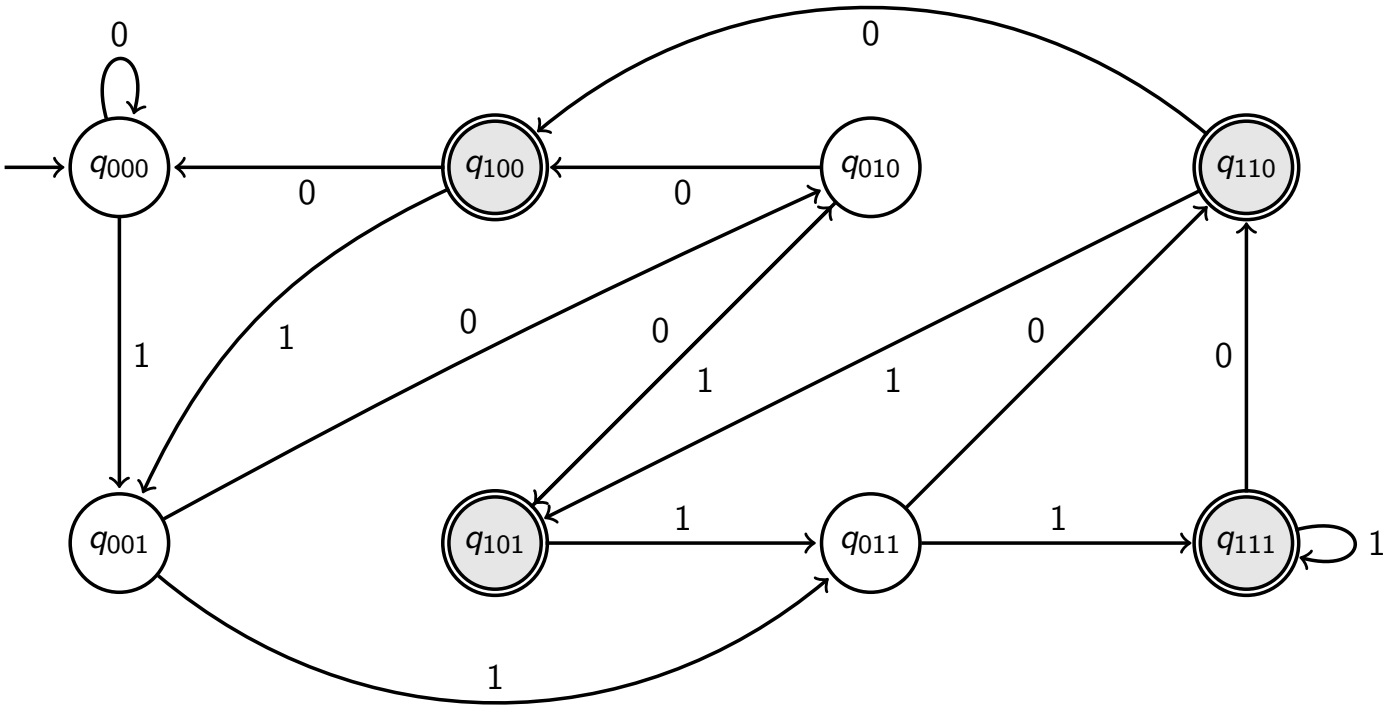


AFNs podem ser mais simples de planejar

- Além de ocupar mais espaço, **entender** o funcionamento desse AFD é muito mais complicado.

Autômatos finitos não-determinísticos

AFD equivalente à N_8:



AFNs podem ser mais simples de planejar

- Além de ocupar mais espaço, **entender** o funcionamento desse AFD é muito mais complicado.

Roteiro

Definição formal de um AFN

Definição

Um autômato finito não-determinístico (AFN) é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, em que

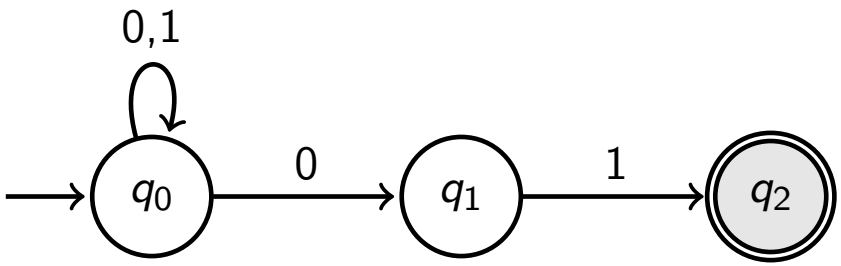
1. Q é um conjunto finito de estados;
2. Σ é o alfabeto da cadeia de entrada;
3. $\delta : Q \times \Sigma \to 2^Q$ é a função de transição que mapeia $Q \times \Sigma$ em 2^Q.
   - $\delta(p, a) = \{q_1, q_2, \ldots, q_r\}$
4. $q_0 \in Q$ é o estado inicial; e
5. $F \subseteq Q$ é o conjunto de estados de aceitação.

Relembrando: 2^Q é o conjunto das partes de Q (o conjunto de todos os subconjuntos).

Definição formal de AFD

Função transição

- $\delta(p, a) = \{q_1, q_2, \ldots, q_r\}$ então quando o AFN está no estado p e lê o símbolo 'a', os próximos estados serão $\{q_1, q_2, \ldots, q_r\}$.



$\delta(q_0, a) = \{q_0, q_1\}$
$\delta(q_1, a) = \{q_2\}$

	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
$\star q_2$	$\varnothing$	$\varnothing$

Tabela: Transições

Função de transição estendida

Definição

A função de transição estendida (ou programa) de N, denotada por

$$\delta^* : Q \times \Sigma^* \to 2^Q$$

é a função $\delta : Q \times \Sigma \to 2^Q$ estendida para palavras definida como:

$$\begin{aligned} \delta^*(q, \mathcal{E}) &= \{q\} \\ \delta^*(q, wa) &= \{p \mid \text{para algum } r \in \delta^*(q, w), p \text{ está em } \delta(r, a)\} \end{aligned}$$

ou seja, é a sucessiva aplicação da função de transição para cada símbolo de w.

Na segunda condição, ao processarmos wa, podemos chegar em p **se e somente se** ao processar w, chegamos em r e podemos ir *de r para q*.

Função programa estendida

continuação. . .

A função programa também pode ser estendida para conjuntos de estados $P \in Q$, com argumentos em $2^Q \times \Sigma^*$:

$$\delta^*(P, w) = \bigcup_{(q \in P)} \delta^*(q, w)$$

para cada $P \subseteq Q$

em outras palavras:

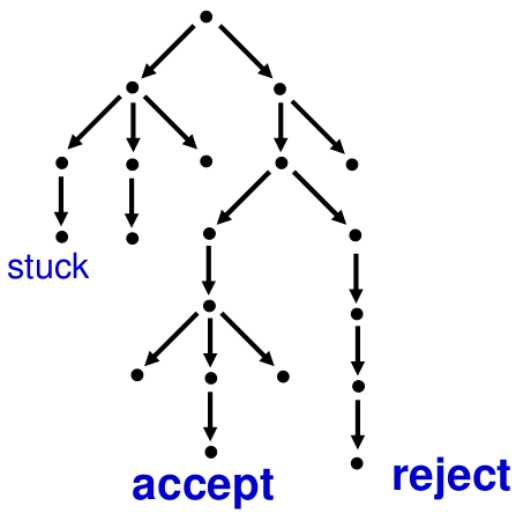
$$\delta^*(\{q_1, q_2, \ldots, q_r\}, w) = \delta^*(q_1, w) \cup \delta^*(q_2, w) \cup \ldots \cup \delta^*(q_r, w)$$

é a união de todas as possibilidades de aplicação da função programa sobre w.

Funcionamento de um AFN

Condições de parada: após processar o último símbolo.

- **aceita** a entrada:
  - se existir pelo menos um caminho, partindo de q_0, que leve N até um estado de aceitação.
- **rejeita** a entrada:
  - se não existir nenhum caminho, partindo de q_0, que leve N até um estado de aceitação.



Linguagem reconhecida

Definição

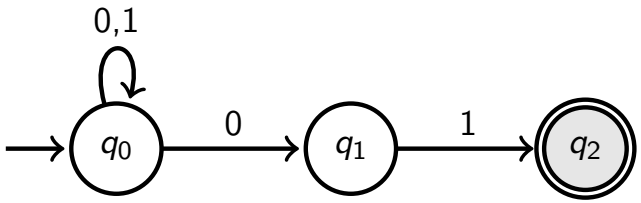
A linguagem reconhecida pelo AFN $N = (Q, \Sigma, \delta, q_0, F)$:

$$L(N) = \{w \mid w \in \Sigma^* \text{ e } \delta^*(q_0, w) \text{ contém um estado em } F\}$$

De forma análoga, w é aceita por M se $\delta^*(q_0, w) \cap F \neq \varnothing$

Autômatos finitos não-determinísticos

Considere N_8 e a cadeia de entrada $w = 01001$



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

então

$$\delta^*(q_0, 01) = \delta^*(\delta(q_0, 0), 1) = \delta^*(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0, q_2\}$$

da mesma forma...

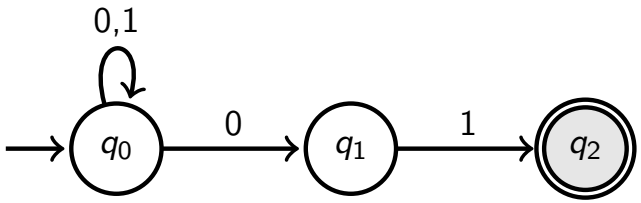
$$\delta(q_0, 010) = \{q_0, q_1\}, \delta(q_0, 0100) = \{q_0, q_1\}$$

e

$$\delta(q_0, 01001) = \{q_0, q_2\}$$

Autômatos finitos não-determinísticos

Considere N_8 e a cadeia de entrada $w = 01001$



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

então

$$\delta^*(q_0, 01) = \delta^*(\delta(q_0, 0), 1) = \delta^*(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0, q_2\}$$

da mesma forma...

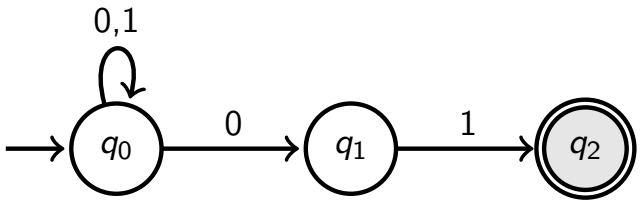
$$\delta(q_0, 010) = \{q_0, q_1\}, \delta(q_0, 0100) = \{q_0, q_1\}$$

e

$$\delta(q_0, 01001) = \{q_0, q_2\}$$

Autômatos finitos não-determinísticos

Considere N_8 e a cadeia de entrada $w = 01001$



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

então

$$\delta^*(q_0, 01) = \delta^*(\delta(q_0, 0), 1) = \delta^*(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0, q_2\}$$

da mesma forma...

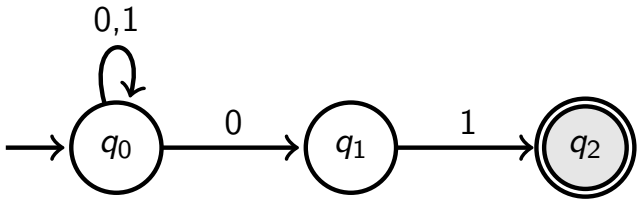
$$\delta(q_0, 010) = \{q_0, q_1\}, \delta(q_0, 0100) = \{q_0, q_1\}$$

e

$$\delta(q_0, 01001) = \{q_0, q_2\}$$

Autômatos finitos não-determinísticos

Considere N_8 e a cadeia de entrada $w = 01001$



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

então

$$\delta^*(q_0, 01) = \delta^*(\delta(q_0, 0), 1) = \delta^*(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0, q_2\}$$

da mesma forma...

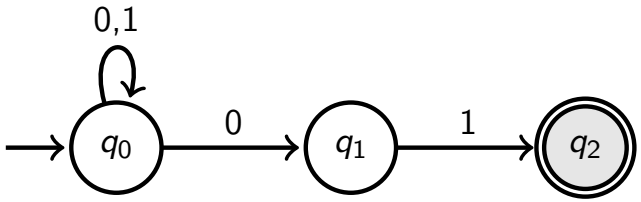
$$\delta(q_0, 010) = \{q_0, q_1\}, \delta(q_0, 0100) = \{q_0, q_1\}$$

e

$$\delta(q_0, 01001) = \{q_0, q_2\}$$

Autômatos finitos não-determinísticos

Considere N_8 e a cadeia de entrada $w = 01001$



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

então

$$\delta^*(q_0, 01) = \delta^*(\delta(q_0, 0), 1) = \delta^*(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0, q_2\}$$

da mesma forma...

$$\delta(q_0, 010) = \{q_0, q_1\}, \delta(q_0, 0100) = \{q_0, q_1\}$$

e

$$\delta(q_0, 01001) = \{q_0, q_2\}$$

Roteiro

Equivalência entre AFDs e AFNs

Vamos ver que a classe de linguagens aceitas[1] pelos AFNs é a mesma aceita pelos Autômatos Finitos Determinísticos (AFDs).

- O não-determinísmo não adiciona poder computacional aos AFs.

Vantagens dos AFNs:

- Podem ser mais fáceis de projetar.
- Em geral, são mais sucintos (menor espaço).
- Úteis para provar teoremas.

[1] Linguagens Regulares

Equivalência entre AFDs e AFNs

Vamos ver que a classe de linguagens aceitas[1] pelos AFNs é a mesma aceita pelos Autômatos Finitos Determinísticos (AFDs).

- O não-determinísmo não adiciona poder computacional aos AFs.

Vantagens dos AFNs:

- Podem ser mais fáceis de projetar.
- Em geral, são mais sucintos (menor espaço).
- Úteis para provar teoremas.

[1] Linguagens Regulares

Equivalência entre AFDs e AFNs

Vamos ver que a classe de linguagens aceitas[1] pelos AFNs é a mesma aceita pelos Autômatos Finitos Determinísticos (AFDs).

- O não-determinísmo não adiciona poder computacional aos AFs.

Vantagens dos AFNs:

- Podem ser mais fáceis de projetar.
- Em geral, são mais sucintos (menor espaço).
- Úteis para provar teoremas.

[1] Linguagens Regulares

Equivalência entre AFDs e AFNs

Vamos ver que a classe de linguagens aceitas[1] pelos AFNs é a mesma aceita pelos Autômatos Finitos Determinísticos (AFDs).

- O não-determinísmo não adiciona poder computacional aos AFs.

Vantagens dos AFNs:

- Podem ser mais fáceis de projetar.
- Em geral, são mais sucintos (menor espaço).
- Úteis para provar teoremas.

[1] Linguagens Regulares

Equivalência entre AFDs e AFNs

Vamos ver que a classe de linguagens aceitas[1] pelos AFNs é a mesma aceita pelos Autômatos Finitos Determinísticos (AFDs).

- O não-determinísmo não adiciona poder computacional aos AFs.

Vantagens dos AFNs:

- Podem ser mais fáceis de projetar.
- Em geral, são mais sucintos (menor espaço).
- Úteis para provar teoremas.

[1] Linguagens Regulares

Equivalência entre AFDs e AFNs

Teorema:

Qualquer linguagem reconhecida por um AFN N também pode ser reconhecida por um AFD M equivalente.

$$L(N) = L(M)$$

Equivalência entre AFDs e AFNs

Equivalência

- AFD $\rightarrow$ AFN.
  - Não precisa ser provado, basta $\delta_D(q, a) = p \rightarrow \delta_N(q, a) = \{p\}$.
- AFN $\rightarrow$ AFD.
  - Prova por construção (iremos apresentar um procedimento).

A ideia central é calcular um AFD M' com estados que *simulem* as diversas combinações de estados alternativos no AFN N.

Equivalência entre AFDs e AFNs

Equivalência

- AFD $\rightarrow$ AFN.
  - Não precisa ser provado, basta $\delta_D(q, a) = p \rightarrow \delta_N(q, a) = \{p\}$.
- AFN $\rightarrow$ AFD.
  - Prova por construção (iremos apresentar um procedimento).

A ideia central é calcular um AFD M' com estados que *simulem* as diversas combinações de estados alternativos no AFN N.

Equivalência entre AFDs e AFNs

Equivalência

- AFD $\rightarrow$ AFN.
  - Não precisa ser provado, basta $\delta_D(q, a) = p \rightarrow \delta_N(q, a) = \{p\}$.
- AFN $\rightarrow$ AFD.
  - Prova por construção (iremos apresentar um procedimento).

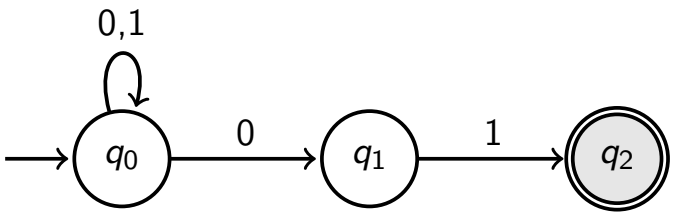
A ideia central é calcular um AFD M' com estados que *simulem* as diversas combinações de estados alternativos no AFN N.

Equivalência entre AFDs e AFNs

Vamos definir o AFD equivalente $M = (\mathbf{Q_D}, \Sigma, \delta_\mathbf{D}, \mathbf{q_0'}, \mathbf{F_D})$, tal que:

- $Q_D = 2^Q$, todos os subconjuntos de Q
  - $q_{ij\ldots k} \in Q_D$ representa o subconjunto $\{q_i, q_j, \ldots, q_k\}$
- $q_0' = q_0$ é o estado inicial.
- F_D é o conjunto de estados que contém um estado final do AFN N.
- $\delta_D(p_{ij\ldots k}, a) = q_{xy\ldots z} \iff$
  $\delta_N^*(\{p_i, p_j, \ldots, p_k\}, a) = \{q_x, q_y, \ldots, q_z\}$.

Exemplo:



	0	1
$\rightarrow \{q_0\}$		
$\{q_1\}$		
$\{q_2\}$		
$\{q_0, q_1\}$		
$\star\{q_0, q_2\}$		
$\star\{q_1, q_2\}$		
$\star\{q_0, q_1, q_2\}$		

Tabela: Transições

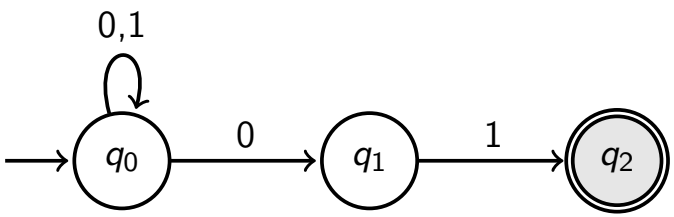
$|Q_D| = 2^n - 1$ (em geral, muitos serão descartados)

Equivalência entre AFDs e AFNs

Vamos definir o AFD equivalente $M = (\mathbf{Q_D}, \Sigma, \delta_\mathbf{D}, \mathbf{q'_0}, \mathbf{F_D})$, tal que:

- $Q_D = 2^Q$, todos os subconjuntos de Q
  - $q_{ij\ldots k} \in Q_D$ representa o subconjunto $\{q_i, q_j, \ldots, q_k\}$
- $q'_0 = q_0$ é o estado inicial.
- F_D é o conjunto de estados que contém um estado final do AFN N.
- $\delta_D(p_{ij\ldots k}, a) = q_{xy\ldots z} \iff$
  $\delta^*_N(\{p_i, p_j, \ldots, p_k\}, a) = \{q_x, q_y, \ldots, q_z\}$.

Exemplo:



	0	1
$\rightarrow q_0$		
q_1		
$\star q_2$		
q_{01}		
$\star q_{02}$		
$\star q_{12}$		
$\star q_{012}$		

Tabela: Transições

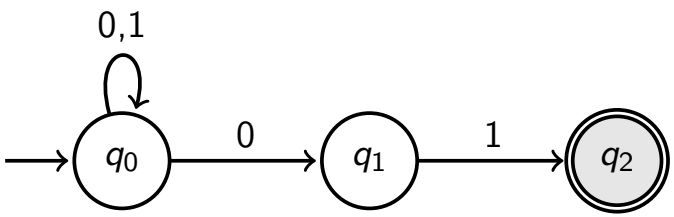
$|Q_D| = 2^n - 1$ (em geral, muitos serão descartados)

Equivalência entre AFDs e AFNs

Vamos definir o AFD equivalente $M = (\mathbf{Q_D}, \Sigma, \delta_\mathbf{D}, \mathbf{q_0'}, \mathbf{F_D})$, tal que:

- $Q_D = 2^Q$, todos os subconjuntos de Q
  - $q_{ij\ldots k} \in Q_D$ representa o subconjunto $\{q_i, q_j, \ldots, q_k\}$
- $q_0' = q_0$ é o estado inicial.
- F_D é o conjunto de estados que contém um estado final do AFN N.
- $\delta_D(p_{ij\ldots k}, a) = q_{xy\ldots z} \iff$
  $\delta_N^*(\{p_i, p_j, \ldots, p_k\}, a) = \{q_x, q_y, \ldots, q_z\}$.

Exemplo:



	0	1
$\rightarrow q_0$		
q_1		
$\star q_2$		
q_{01}		
$\star q_{02}$		
$\star q_{12}$		
$\star q_{012}$		

Tabela: Transições

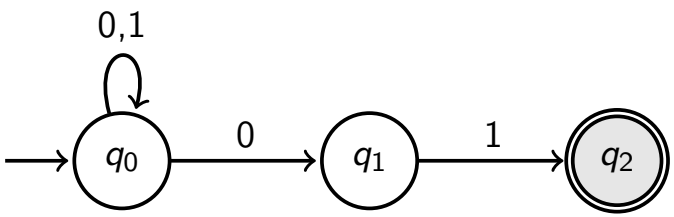
$|Q_D| = 2^n - 1$ (em geral, muitos serão descartados)

Equivalência entre AFDs e AFNs

Vamos definir o AFD equivalente $M = (\mathbf{Q_D}, \Sigma, \delta_\mathbf{D}, \mathbf{q_0'}, \mathbf{F_D})$, tal que:

- $Q_D = 2^Q$, todos os subconjuntos de Q
  - $q_{ij\ldots k} \in Q_D$ representa o subconjunto $\{q_i, q_j, \ldots, q_k\}$
- $q_0' = q_0$ é o estado inicial.
- F_D é o conjunto de estados que contém um estado final do AFN N.
- $\delta_D(p_{ij\ldots k}, a) = q_{xy\ldots z} \iff$
  $\delta_N^*(\{p_i, p_j, \ldots, p_k\}, a) = \{q_x, q_y, \ldots, q_z\}$.

Exemplo:



	0	1
$\rightarrow q_0$		
q_1		
$\star q_2$		
q_{01}		
$\star q_{02}$		
$\star q_{12}$		
$\star q_{012}$		

Tabela: Transições

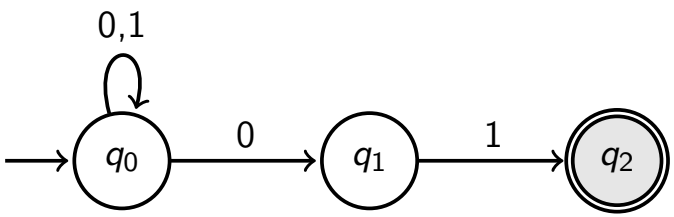
$|Q_D| = 2^n - 1$ (em geral, muitos serão descartados)

Equivalência entre AFDs e AFNs

Vamos definir o AFD equivalente $M = (\mathbf{Q_D}, \Sigma, \delta_\mathbf{D}, \mathbf{q'_0}, \mathbf{F_D})$, tal que:

- $Q_D = 2^Q$, todos os subconjuntos de Q
  - $q_{ij\ldots k} \in Q_D$ representa o subconjunto $\{q_i, q_j, \ldots, q_k\}$
- $q'_0 = q_0$ é o estado inicial.
- F_D é o conjunto de estados que contém um estado final do AFN N.
- $\delta_D(p_{ij\ldots k}, a) = q_{xy\ldots z} \iff$
  $\delta^*_N(\{p_i, p_j, \ldots, p_k\}, a) = \{q_x, q_y, \ldots, q_z\}$.

Exemplo:



	0	1
$\rightarrow q_0$		
q_1		
$\star q_2$		
q_{01}		
$\star q_{02}$		
$\star q_{12}$		
$\star q_{012}$		

Tabela: Transições

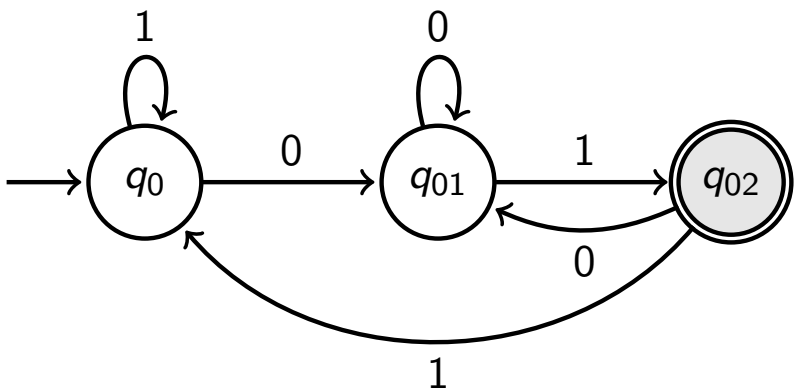
$|Q_D| = 2^n - 1$ (em geral, muitos serão descartados)

Equivalência entre AFDs e AFNs

Resultado:

	0	1
$\rightarrow q_0$	q_{01}	q_0
q_1	$\perp$	q_2
q_2	$\perp$	$\perp$
q_{01}	q_{01}	q_{02}
$*q_{02}$	q_{01}	q_0
$*q_{12}$	$\perp$	q_2
$*q_{012}$	q_{01}	q_{02}

Tabela: Transições



Equivalência entre AFDs e AFNs

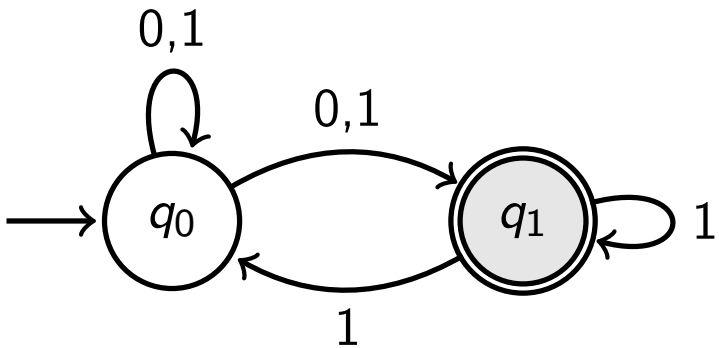
Na prática, muitos estados no AFD equivalente não são acessíveis a partir de q_0:

- Uma boa prática é inserir apenas estados acessíveis partindo de q_0, criando apenas estados *alcançaveis*

Equivalência entre AFDs e AFNs

Outro exemplo:

- Seja $N_9 = (Q = \{q_0, q_1\}, \Sigma = \{0, 1\}, \delta, q_0, \{q_1\}$ um AFN:



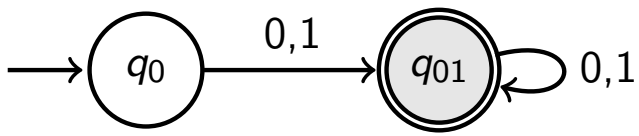
- Vamos contruir $M_9 = (Q_D, \Sigma, \delta_D, q_0', F_D)$, tal que $L(N_9) = L(M_9)$

Equivalência entre AFDs e AFNs

Resultado:

	0	1
$\rightarrow q_0$	q_{01}	q_{01}
$\star q_1$	$\perp$	q_{01}
$\star q_{01}$	q_{01}	q_{01}

Tabela: Transições



Equivalência entre AFDs e AFNs

O AFD M simula todas as computações do AFN N?

Prova formal:

É fácil ver por indução no tamanho de $w = w_1 w_2 \ldots w_n$ que

$$\delta_D(q_0', w) = q_{ij\ldots k}$$

se, e somente se

$$\delta_N(\{q_0\}, w) = \{q_i, q_j, \ldots, q_k\}$$

Então, $\delta_D(q_0', w)$ está em F_D quando $\delta_N(q_0, w)$ contém um estado $q \in F$.

Portanto:

$$L(D) = L(N)$$

Equivalência entre AFDs e AFNs

O AFD M simula todas as computações do AFN N?

Prova formal:

É fácil ver por indução no tamanho de $w = w_1 w_2 \ldots w_n$ que

$$\delta_D(q_0', w) = q_{ij\ldots k}$$

se, e somente se

$$\delta_N(\{q_0\}, w) = \{q_i, q_j, \ldots, q_k\}$$

Então, $\delta_D(q_0', w)$ está em F_D quando $\delta_N(q_0, w)$ contém um estado $q \in F$.

Portanto:

$$L(D) = L(N)$$

Equivalência entre AFDs e AFNs

O AFD M simula todas as computações do AFN N?

Prova formal:

É fácil ver por indução no tamanho de $w = w_1 w_2 \ldots w_n$ que

$$\delta_D(q_0', w) = q_{ij\ldots k}$$

se, e somente se

$$\delta_N(\{q_0\}, w) = \{q_i, q_j, \ldots, q_k\}$$

Então, $\delta_D(q_0', w)$ está em F_D quando $\delta_N(q_0, w)$ contém um estado $q \in F$.

- Portanto:

$$L(M) = L(N)$$

Equivalência entre AFDs e AFNs

Base da indução: $|w| = 0$, portanto $w = \mathcal{E}$

$$\delta^*_D(q'_0, \mathcal{E}) = q_0 \text{ sse } \delta^*_N(\{q_0\}, w) = \{q_0\}$$

- Verdadeiro, por definição.

Hipótese de indução: $|w| = n$ e $n \geq 1$, suponha que:

$$\delta^*_D(q'_0, w) = q_{uv\ldots w} \text{ sse } \delta^*_N(\{q_0\}, w) = \{q_u, q_v, \ldots, q_w\}$$

Passo: $|wa| = n + 1$ e $n \geq 1$:

$$\delta^*_D(q'_0, wa) = \delta^*_D(\delta^*_D(q'_0, w), a)$$

Equivalência entre AFDs e AFNs

Base da indução: $|w| = 0$, portanto $w = \mathcal{E}$

$$\delta_D^*(q_0', \mathcal{E}) = q_0 \text{ sse } \delta_N^*(\{q_0\}, w) = \{q_0\}$$

- Verdadeiro, por definição.

Hipótese de indução: $|w| = n$ e $n \geq 1$, suponha que:

$$\delta_D^*(q_0', w) = q_{uv\ldots w} \text{ sse } \delta_N^*(\{q_0\}, w) = \{q_u, q_v, \ldots, q_w\}$$

Passo: $|wa| = n + 1$ e $n \geq 1$:

$$\delta_D^*(q_0', wa) = \delta_D^*(\delta_D^*(q_0', w), a)$$

Equivalência entre AFDs e AFNs

Base da indução: $|w| = 0$, portanto $w = \mathcal{E}$

$$\delta_D^*(q_0', \mathcal{E}) = q_0 \text{ sse } \delta_N^*(\{q_0\}, w) = \{q_0\}$$

- Verdadeiro, por definição.

Hipótese de indução: $|w| = n$ e $n \geq 1$, suponha que:

$$\delta_D^*(q_0', w) = q_{uv\ldots w} \text{ sse } \delta_N^*(\{q_0\}, w) = \{q_u, q_v, \ldots, q_w\}$$

Passo: $|wa| = n + 1$ e $n \geq 1$:

$$\delta_D^*(q_0', wa) = \delta_D^*(\delta_D^*(q_0', w), a)$$

Equivalência entre AFDs e AFNs

Passo: $|wa| = n + 1$ e $n \geq 1$:

$$\delta^*_D(q'_0, wa) = \delta^*_D(\delta^*_D(q'_0, w), a)$$

Pela HI:

$$\delta^*_D(q'_0, w) = q_{uv\ldots w} \text{ sse } \delta^*_N(\{q_0\}, w) = \{q_u, q_v, \ldots, q_w\}$$

Pela definição:

$$\delta_D(q_{uv\ldots w}, a) = q_{ij\ldots k} \text{ sse } \delta_N(\{q_u, q_v, \ldots, q_w\}, a) = \{q_i, q_j, \ldots, q_k\}$$

Então:

$$\delta^*_D(q'_0, wa) = q_{ij\ldots k} \text{ sse } \delta^*_N(q_0, wa) = \{q_i, q_j, \ldots, q_k\}$$

- Se $q \in \{q_i, q_j, \ldots, q_k\}$ for estado de aceitação, então $q_{ij\ldots k}$ também é.

Portanto:

- $L(M) = L(N)$

□

Equivalência entre AFDs e AFNs

Passo: $|wa| = n + 1$ e $n \geq 1$:

$$\delta_D^*(q_0', wa) = \delta_D^*(\delta_D^*(q_0', w), a)$$

Pela HI:

$$\delta_D^*(q_0', w) = q_{uv\ldots w} \text{ sse } \delta_N^*(\{q_0\}, w) = \{q_u, q_v, \ldots, q_w\}$$

Pela definição:

$$\delta_D(q_{uv\ldots w}, a) = q_{ij\ldots k} \text{ sse } \delta_N(\{q_u, q_v, \ldots, q_w\}, a) = \{q_i, q_j, \ldots, q_k\}$$

Então:

$$\delta_D^*(q_0', wa) = q_{ij\ldots k} \text{ sse } \delta_N^*(q_0, wa) = \{q_i, q_j, \ldots, q_k\}$$

- Se $q \in \{q_i, q_j, \ldots, q_k\}$ for estado de aceitação, então $q_{ij\ldots k}$ também é.

Portanto:

- $L(M) = L(N)$

□

Equivalência entre AFDs e AFNs

Passo: $|wa| = n + 1$ e $n \geq 1$:

$$\delta_D^*(q_0', wa) = \delta_D^*(\delta_D^*(q_0', w), a)$$

Pela HI:

$$\delta_D^*(q_0', w) = q_{uv\ldots w} \text{ sse } \delta_N^*(\{q_0\}, w) = \{q_u, q_v, \ldots, q_w\}$$

Pela definição:

$$\delta_D(q_{uv\ldots w}, a) = q_{ij\ldots k} \text{ sse } \delta_N(\{q_u, q_v, \ldots, q_w\}, a) = \{q_i, q_j, \ldots, q_k\}$$

Então:

$$\delta_D^*(q_0', wa) = q_{ij\ldots k} \text{ sse } \delta_N^*(q_0, wa) = \{q_i, q_j, \ldots, q_k\}$$

- Se $q \in \{q_i, q_j, \ldots, q_k\}$ for estado de aceitação, então $q_{ij\ldots k}$ também é.

Portanto:

- $L(M) = L(N)$

□

Equivalência entre AFDs e AFNs

Passo: $|wa| = n + 1$ e $n \geq 1$:

$$\delta_D^*(q_0', wa) = \delta_D^*(\delta_D^*(q_0', w), a)$$

Pela HI:

$$\delta_D^*(q_0', w) = q_{uv\ldots w} \text{ sse } \delta_N^*(\{q_0\}, w) = \{q_u, q_v, \ldots, q_w\}$$

Pela definição:

$$\delta_D(q_{uv\ldots w}, a) = q_{ij\ldots k} \text{ sse } \delta_N(\{q_u, q_v, \ldots, q_w\}, a) = \{q_i, q_j, \ldots, q_k\}$$

Então:

$$\delta_D^*(q_0', wa) = q_{ij\ldots k} \text{ sse } \delta_N^*(q_0, wa) = \{q_i, q_j, \ldots, q_k\}$$

- Se $q \in \{q_i, q_j, \ldots, q_k\}$ for estado de aceitação, então $q_{ij\ldots k}$ também é.

Portanto:

- $L(M) = L(N)$

Equivalência entre AFDs e AFNs

Passo: $|wa| = n + 1$ e $n \geq 1$:

$$\delta_D^*(q_0', wa) = \delta_D^*(\delta_D^*(q_0', w), a)$$

Pela HI:

$$\delta_D^*(q_0', w) = q_{uv\ldots w} \text{ sse } \delta_N^*(\{q_0\}, w) = \{q_u, q_v, \ldots, q_w\}$$

Pela definição:

$$\delta_D(q_{uv\ldots w}, a) = q_{ij\ldots k} \text{ sse } \delta_N(\{q_u, q_v, \ldots, q_w\}, a) = \{q_i, q_j, \ldots, q_k\}$$

Então:

$$\delta_D^*(q_0', wa) = q_{ij\ldots k} \text{ sse } \delta_N^*(q_0, wa) = \{q_i, q_j, \ldots, q_k\}$$

- Se $q \in \{q_i, q_j, \ldots, q_k\}$ for estado de aceitação, então $q_{ij\ldots k}$ também é.

Portanto:

- $L(M) = L(N)$

□

Fim

Dúvidas?

Roteiro

Referências

Referências:

1. *"Introdução à Teoria da Computação"* de M. Sipser, 2007.
2. *"Introdução à Teoria de Autômatos, Linguagens e Computação"* de J. E. Hopcroft, R. Motwani, e J. D. Ullman, 2003.
3. Materiais adaptados dos slides do Prof. Evandro E. S. Ruiz, da USP.