Introdução
Autômatos Finitos

Linguagens
Autômatos Finitos Determinísticos
Representações de Autômatos

Tradução dos slides do Prof. Jeffrey D. Ullman (Stanford University)

Alfabetos

- Um *alfabeto* é um conjunto finito de símbolos e não-vazio.
- Exemplos: ASCII, Unicode, {0,1} (*alfabeto binário*), {a,b,c}.
- Usamos o símbolo Σ para um alfabeto.

Strings

- Um *string* (ou palavra ou cadeia) é uma sequência finita de símbolos escolhidos de algum alfabeto Σ.
  - 011 é um string do alfabeto binário $\Sigma=\{0,1\}$
- Σ^* denota o conjunto de todos os strings sobre um alfabeto Σ.
- ϵ denota *string vazio* (string com zero ocorrência de símbolos).

Exemplo: Strings

- {0,1}* = {ϵ, 0, 1, 00, 01, 10, 11, 000, 001, . . . }
- Sutileza: 0 como uma string e 0 como um símbolo
  - O contexto determina o que é.

Linguagens

- Uma *linguagem* é um subconjunto de Σ^* para algum alfabeto Σ.
- Exemplo: O conjunto de strings de 0's e 1's que não tenha dois 1's consecutivos.
- L = {ϵ, 0, 1, 00, 01, 10, 000, 001, 010, 100, 101, 0000, 0001, 0010, 0100, 0101, 1000, 1001, 1010, . . . }

Hmm... 1 de comprimento 0, 2 de comprimento 1, 3, de comprimento 2, 5 de comprimento 3, 8 de comprimento 4. Eu pergunto quantos de comprimento 5?

Autômatos Finitos Determinísticos

- Vamos apresentar a noção formal de um autômato finito;
- O termo “determinístico” se refere ao fato de que, para cada entrada, existe um e somente um estado ao qual o autômato pode transitar a partir de seu estado atual;
- Em contraste, autômatos finitos “não-determinísticos” podem estar em vários estados ao mesmo tempo.

Autômatos Finitos Determinísticos

- Um autômato finito determinístico (DFA) consiste em:
  1. Um conjunto finito de *estados* (Q).
  2. Um conjunto finito de *símbolos de entrada* (Σ).
  3. Uma *função de transição* (δ).
  4. Um *estado inicial* (q_0, em Q).
  5. Um conjunto de *estados finais* ou *de aceitação* ($F \subseteq Q$).

A Função de Transição

- Toma como argumentos um estado e um símbolo de entrada e retorna um estado.
- $\delta(q, a) = p$ (estado para o qual o DFA vai quando se está no estado *q* e *a* de entrada é recebido).

Representação de um DFA

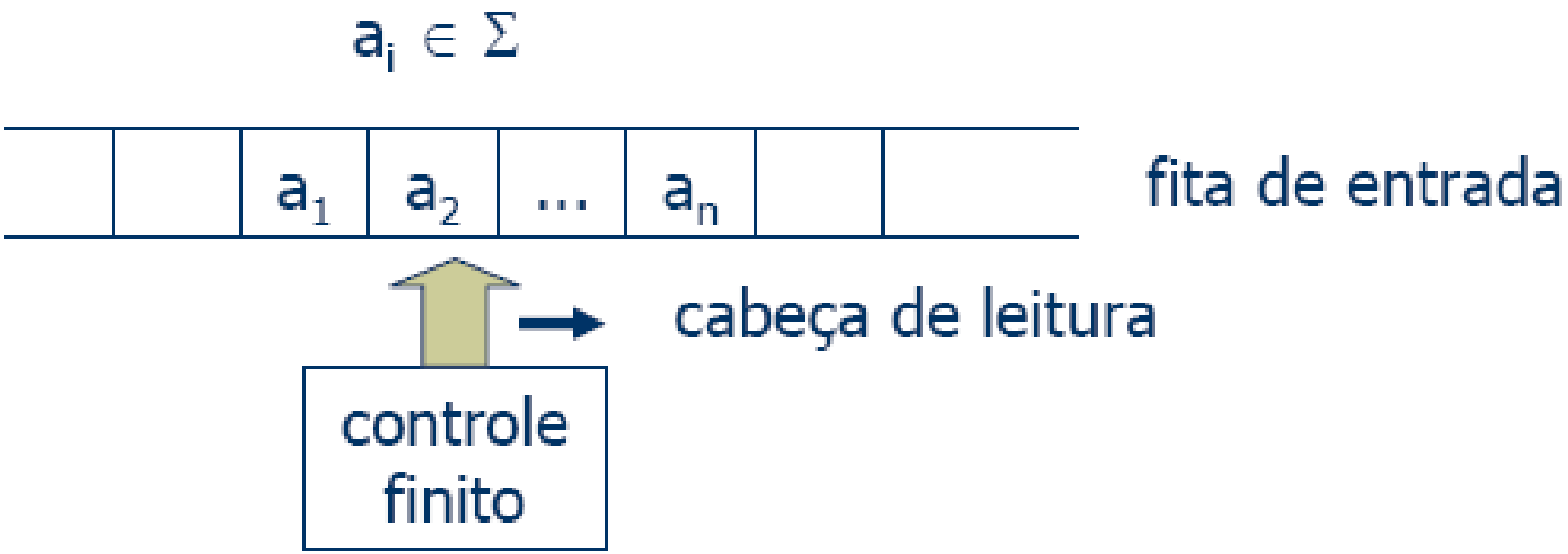
- Um DFA A é representado por uma quíntupla:

$$A = (Q, \Sigma, \delta, q_0, F)$$

Q - conjunto finito não vazio (estados)
Σ - alfabeto (de entrada); $\Sigma \cap Q = \emptyset$
$q_0 \in Q$ (estado inicial)
$F \subseteq Q$ (conjunto de estados finais)
$\delta : Q \times \Sigma \rightarrow Q$ (função de transição de estados)

Como um DFA processa strings

- Simbolicamente:



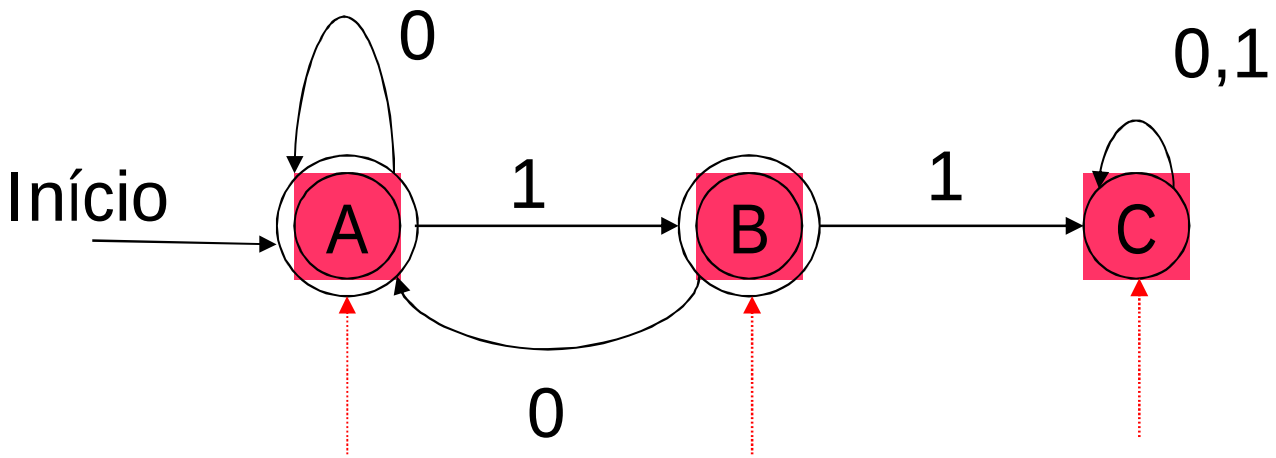
$\delta(q,a) = q'$ autômato estando no estado q e lendo o símbolo a na fita de entrada, move a cabeça leitora uma posição para a direita e vai para o estado q'.

Diagrama de transições de DFA's

- Nós = estados.
- Arcos representa a função de transição.
  - Arco do estado *q* para o estado *p* rotulado por *a* (ou lista de simbolos de *q* para *p*)
- Seta identificada como "Início" para o estado inicial.
- Estados de aceitação são marcados por um circulo duplo.

Exemplo: Grafo de um DFA

Aceita todos os strings sem dois 1's consecutivos.

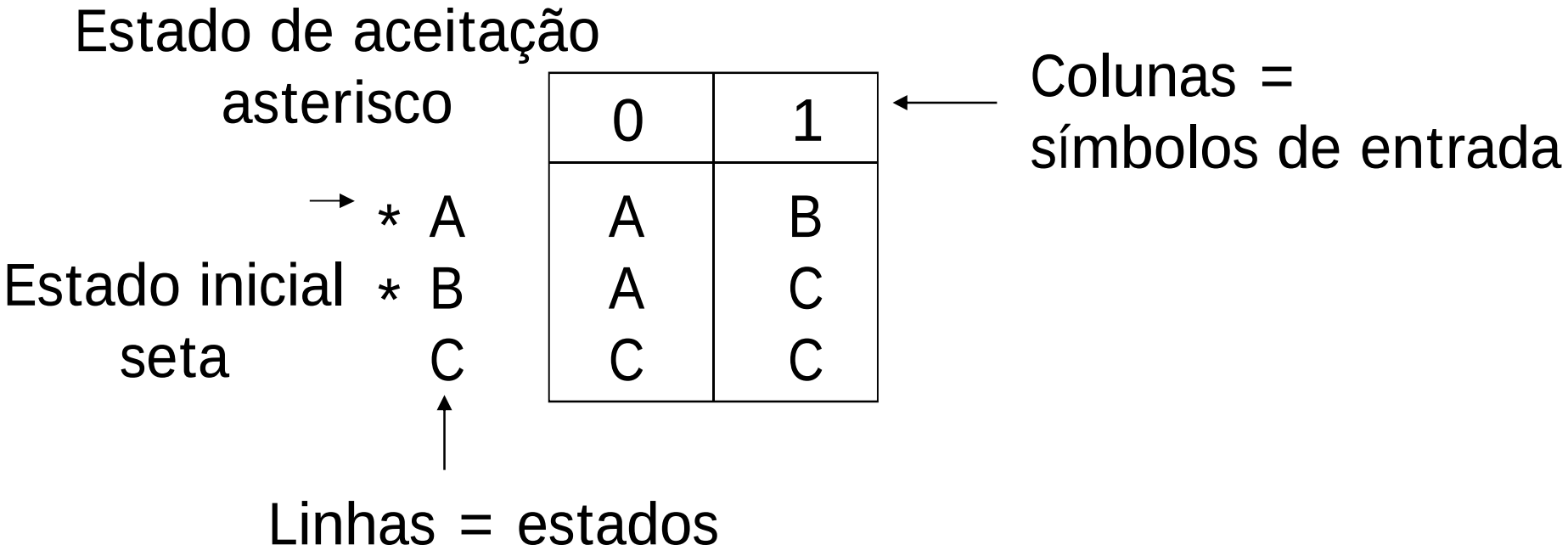


String OK, não termina em 1.

String OK, termina em um único 1.

1's consecutivos foram vistos.

Representação Alternativa: Tabela de Transitição



	0	1
→ * A	A	B
* B	A	C
C	C	C

Função de Transição Extendida

- Linguagem de um DFA: conjunto de rótulos ao longo de todos os caminhos que levam do estado inicial a qualquer estado de aceitação;
- Para tornar exata a noção da linguagem de um DFA

  Função de transição extendida

Função de Transição Extendida

- Descreve o que acontece quando começamos em qualquer estado e seguimos qualquer sequência de entradas.
- Indução sobre o comprimento da string.
- Base: $\delta(q, \epsilon) = q$
- Indução: $\delta(q,wa) = \delta(\delta(q,w),a)$
  - w é uma string; a é um símbolo de entrada.

δ Extendida: Intuição

- Convenção:
  - ... w, x, y, x são strings.
  - a, b, c,... são símbolos.
- δ Extendida é calculado para o estado q e entradas $a_1a_2...a_n$ seguindo um caminho no grafo de transição, a partir de q e selecionando os arcos com rótulos a_1, a_2, ..., a_n, por sua vez.

Delta-chapêu

- No livro, o δ extendida tem um "chapêu" para distinguir de δ.
- $\hat{\delta}(q, a) = \delta(\hat{\delta}(q, \epsilon), a) = \delta(q, a)$

Deltas Extendidas

Exemplo: δ Extendida

	0	1
→ * A	A	B
* B	A	C
C	C	C

$\hat{\delta}(A, w)$

$w = \{011\}$

$\hat{\delta}(A, \epsilon) = A$

$\hat{\delta}(A, 0) = \delta(\hat{\delta}(A, \epsilon), 0) = \delta(A, 0) = A$

$\hat{\delta}(A, 01) = \delta(\hat{\delta}(A, 0), 1) = \delta(A, 1) = B$

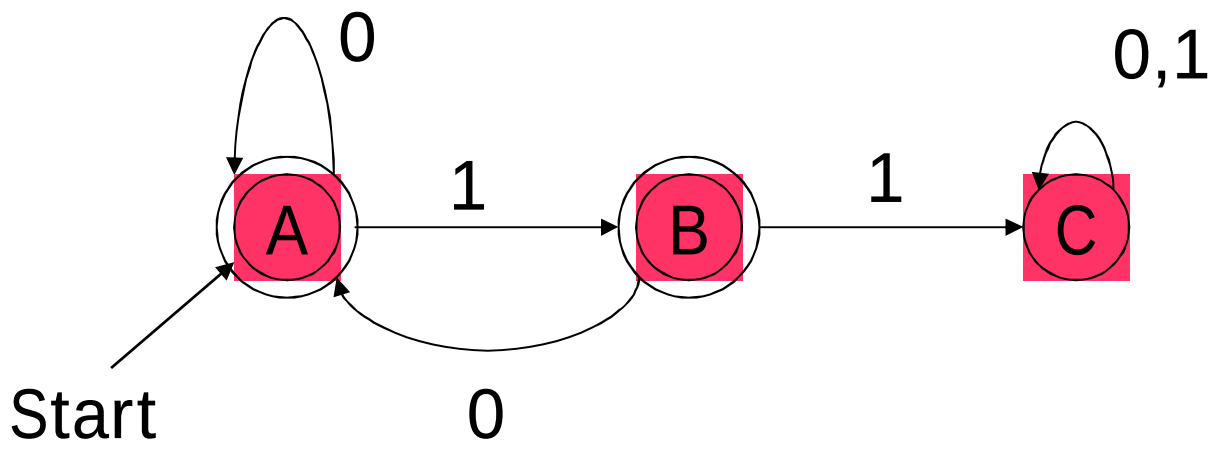
$\hat{\delta}(A, 011) = \delta(\hat{\delta}(A, 01), 1) = \delta(B, 1) = C$

Linguagem de um DFA

- Autômato de todos os tipos definem linguagens.
- Se A é um autômato, L(A) é sua linguagem.
- Para um DFA A, L(A) é o conjunto de strings que levam o estado inicial até um dos estados de aceitação.
- Formalmente: L(A) = o conjunto de strings w tal que $\hat{\delta}(q_0, w)$ está em F.

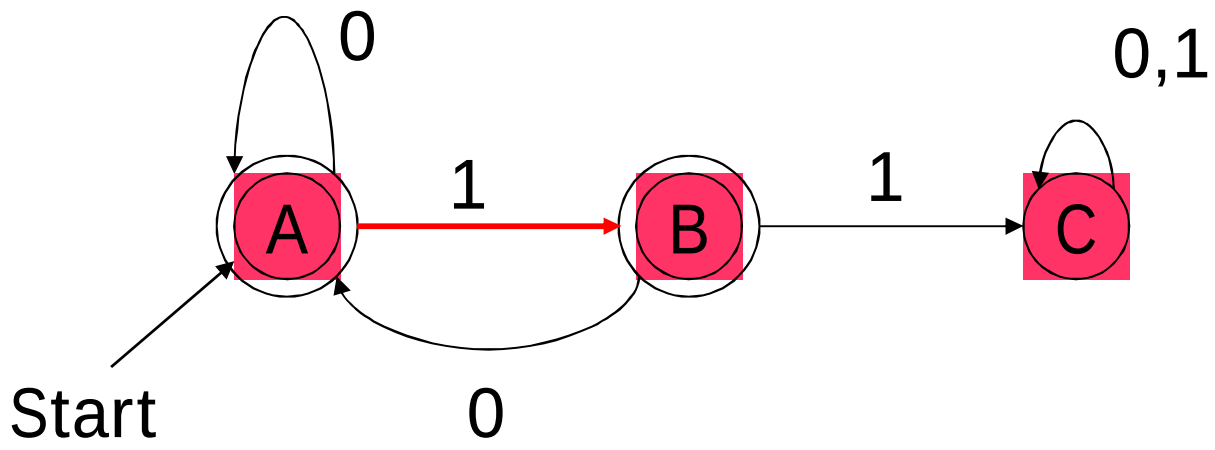
Exemplo: String numa Linguagem

A string 101 está na linguagem do DFA abaixo.
Comece em A.



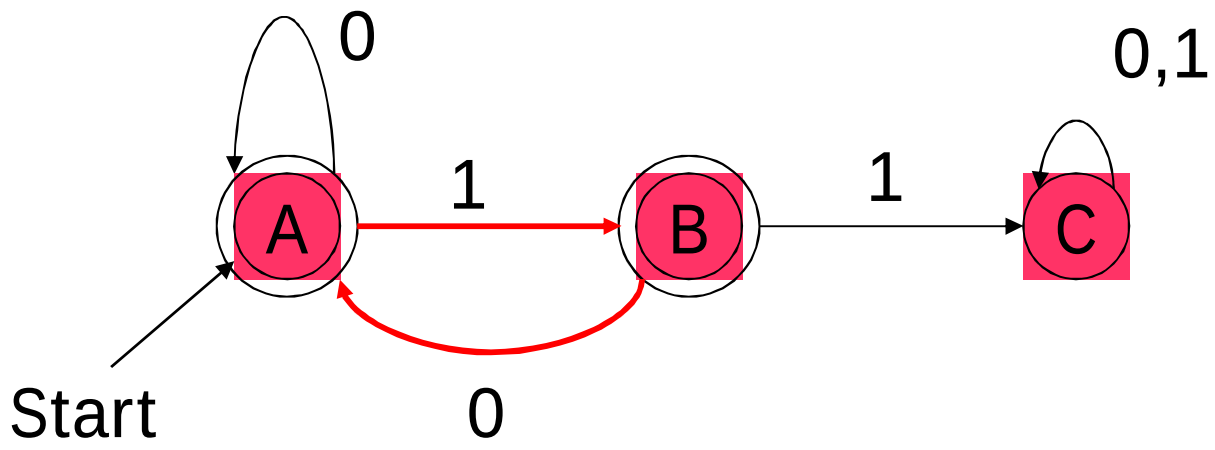
Exemplo: String numa Linguagem

A string 101 está na linguagem do DFA abaixo.
Siga o arco rotulado como 1.



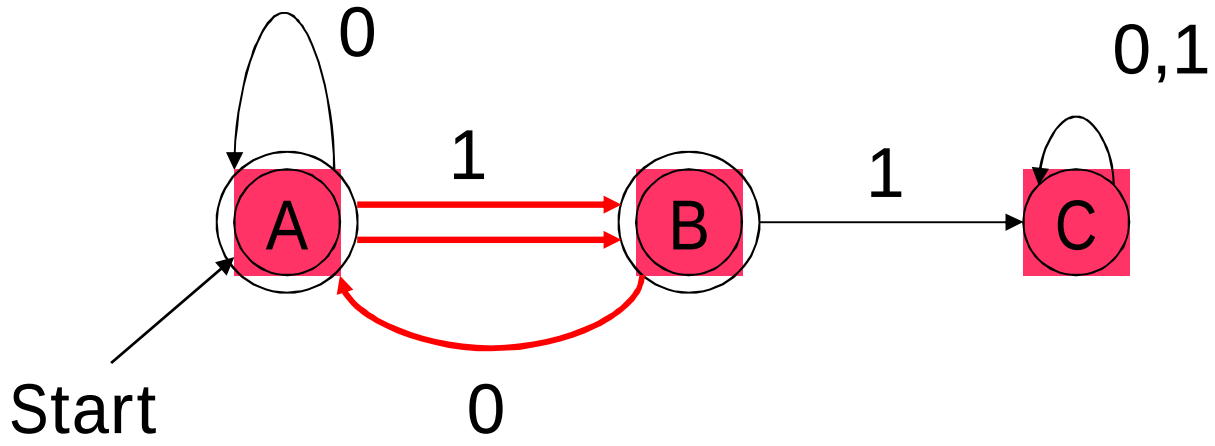
Exemplo: String numa Linguagem

A string 101 está na linguagem do DFA abaixo.
Siga o arco rotulado como 0 do estado corrente B.



Exemplo: String numa Linguagem

A string 101 está na linguagem do DFA abaixo.
Finalmente, siga o arco rotulado como 1 do estado corrente A.



Resultado é um estado de aceitação, então 101 está nesta linguagem.

Exemplo – Concluído

- A linguagem do nosso DFA exemplo é:

{w | w está em {0,1}* e w não tem dois 1's consecutivos}

Tal que...

Estas condições sobre w são verdadeiras.

Leia o *conjunto* como "O conjunto de strings w...

Linguagem Regular

- Uma linguagem L é *regular* se esta é a linguagem aceita por algum DFA.
  - Note: o DFA precisa aceitar somente as strings em L, não outras.
- Algumas linguagems são não-regulares.
  - Intuitivamente, linguagens regulares "não podem contar" para inteiros arbitrariamente elevados.

Exemplo: Uma Linguagem não-regular

$L_1 = \{0^n1^n \mid n \geq 1\}$

- Note: a^i é convencional para i a's.
  - Ex. $0^4 = 0000$.
- Leia: "O conjunto de strings consistindo de n 0's seguidos por n 1's, tal que n é no mínimo 1.
- Assim, $L_1 = \{01, 0011, 000111, \ldots\}$

Outro Exemplo

L_2 = {w | w em {(,)}* e w é *balanceado* }

- Note: alfabeto consiste dos símbolos parentesis ’(’ e ’)’.
- Pares balanceados são aqueles que podem aparecer em expressões aritimeticas.
  - Ex.: (), ()(), (()), (()()),...

Mas muitas Linguagens são Regulares

- Linguagems Regulares podem ser descritas de muitas formas, ex. Expressões regulares.
- Elas aparecem em muitos contextos e tem muitas propriedades úteis.
- Exemplo: as strings que representam números de ponto flutuante na sua linguagem favorita é uma linguagem regular.

Exemplo: Uma Linguagem Regular

L_3 = { w | w em {0,1}* e w, visto com um inteiro binário é divisível por 23}

- O DFA:
  - 23 estados, nomeados 0, 1,...,22.
  - Correspondem aos 23 restos de um inteiro dividido por 23.
  - Estado inicial e de aceitação (único) é 0.

Transições do DFA para L_3

- Se a string w representa o inteiro i, então assume $\delta(0, w) = i\%23$.
- Então w0 representa o inteiro 2i, por isso queremos $\delta(i\%23, 0) = (2i)\%23$.
- Da mesma forma: w1 representa 2i+1, então quermos $\delta(i\%23, 1) = (2i+1)\%23$.
- Exemplo: $\delta(15,0) = 30\%23 = 7$; $\delta(11,1) = 23\%23 = 0$.

Idea-chave: projetar um DFA pensando o que cada estado precisa lembrar sobre o passado.

Outro Exemplo

L_4 = { w | w em {0,1}* e w, visto como o número binário reverso é divisível por 23}

- Exemplo: 01110100 está em L_4, porque seu reverso, 00101110 é 46 em binário.
- Difícil de construir um DFA.
- Mas o teorema diz que o reverso de uma linguagem regular também é regular.