Tema 2

Hierarquia de Chomsky/ Linguagens regulares

Professora:
Ariane Machado Lima

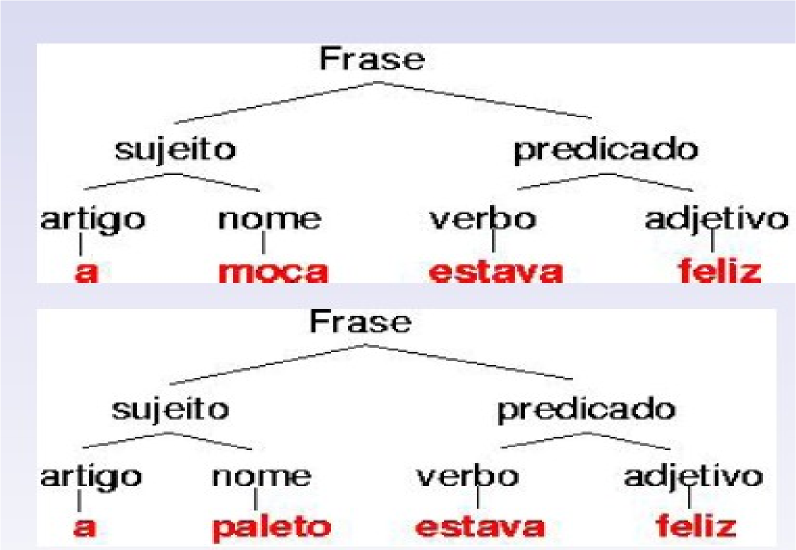
Vídeo 1

Gramáticas - Conceitos Básicos e Hierarquia de Chomsky

Professora:
Ariane Machado Lima

Gramáticas

Frase	$\longrightarrow$	sujeito	predicado
sujeito	$\longrightarrow$	artigo	nome
artigo	$\longrightarrow$	a	
artigo	$\longrightarrow$	o	
nome	$\longrightarrow$	paletó	
nome	$\longrightarrow$	moça	
nome	$\longrightarrow$	dia	
predicado	$\longrightarrow$	verbo	adjetivo
verbo	$\longrightarrow$	é	
verbo	$\longrightarrow$	estava	
adjectivo	$\longrightarrow$	feliz	
adjectivo	$\longrightarrow$	azul	

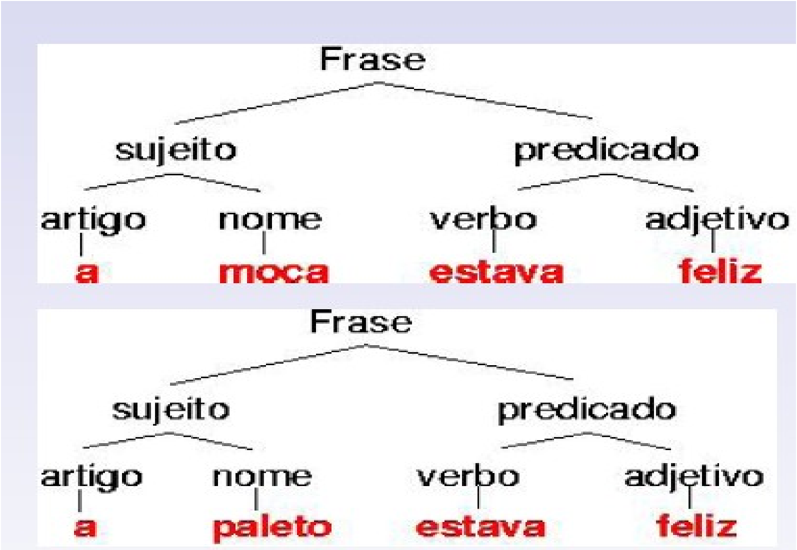


conjunto de produções

Gramáticas

símbolo inicial

Frase	$\longrightarrow$	sujeito	predicado
sujeito	$\longrightarrow$	artigo	nome
artigo	$\longrightarrow$	a	
artigo	$\longrightarrow$	o	
nome	$\longrightarrow$	paletó	
nome	$\longrightarrow$	moça	
nome	$\longrightarrow$	dia	
predicado	$\longrightarrow$	verbo	adjetivo
verbo	$\longrightarrow$	é	
verbo	$\longrightarrow$	estava	
adjectivo	$\longrightarrow$	feliz	
adjectivo	$\longrightarrow$	azul	



símbolos não-terminais

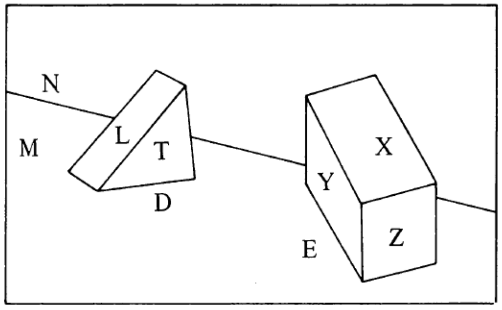
símbolos terminais

Gramáticas

- Uma gramática é capaz de representar um **conjunto de sequências** (cadeias)
- → representa um padrão
- Ex1: sequências de sítios de ligação de um dado fator de transcrição

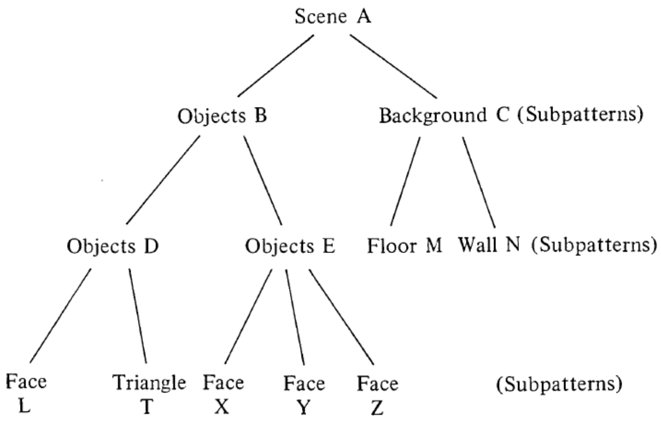
Gramáticas

- Uma gramática é capaz de representar um **conjunto de sequências** (cadeias)
- → representa um padrão
- Ex2: sequências que representam imagens que seguem um padrão



Scene A

a

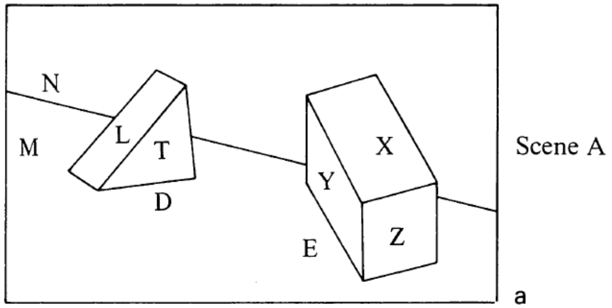


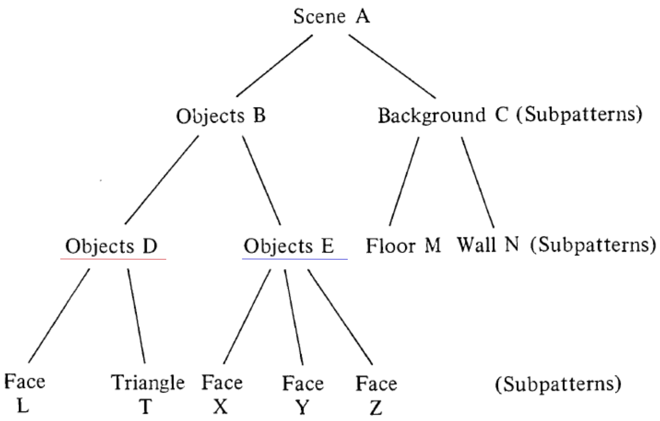
<Scene A> → <Objects B> <Background C>
<Objects B> → <Objects D> <Objects E>
<Objects D> → <Face L> <Triangle T>
<Objects E> → <Face X> <Face Y> <Face Z>
<Background C> → <Floor M> <Wall N>

(((LT)(XYZ))(MN))

Gramáticas

- Uma gramática é capaz de representar um **conjunto de sequências** (cadeias)
- → representa um padrão
- Ex2: sequências que representam imagens que seguem um padrão



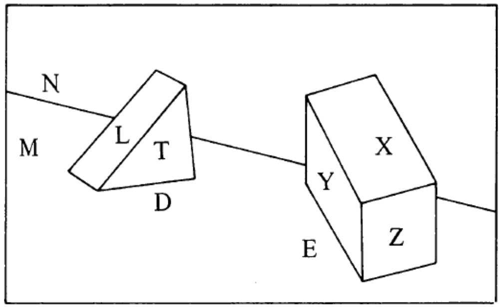


<Scene A> → <Objects B> <Background C>
<Objects B> → <Objects D> <Objects E>
<Objects D> → <Face L> <Triangle T>
<Objects E> → <Face X> <Face Y> <Face Z>
<Background C> → <Floor M> <Wall N>

(((LT)(XYZ))(MN))

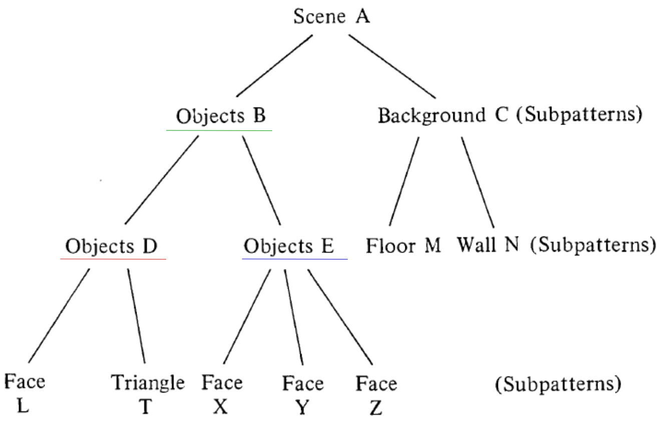
Gramáticas

- Uma gramática é capaz de representar um **conjunto de sequências** (cadeias)
- → representa um padrão
- Ex2: sequências que representam imagens que seguem um padrão



Scene A

a

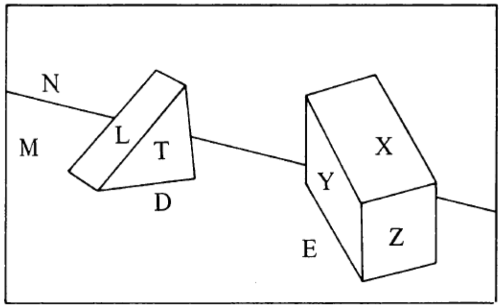


<Scene A> → <Objects B> <Background C>
<Objects B> → <Objects D> <Objects E>
<Objects D> → <Face L> <Triangle T>
<Objects E> → <Face X> <Face Y> <Face Z>
<Background C> → <Floor M> <Wall N>

(((LT)(XYZ))(MN))

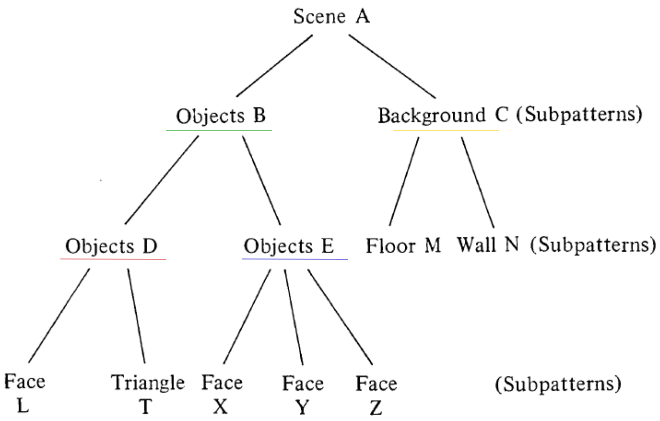
Gramáticas

- Uma gramática é capaz de representar um **conjunto de sequências** (cadeias)
- → representa um padrão
- Ex2: sequências que representam imagens que seguem um padrão



Scene A

a

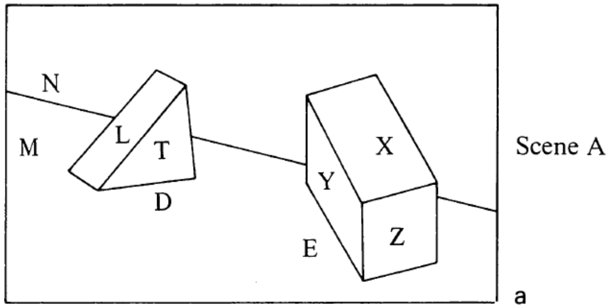


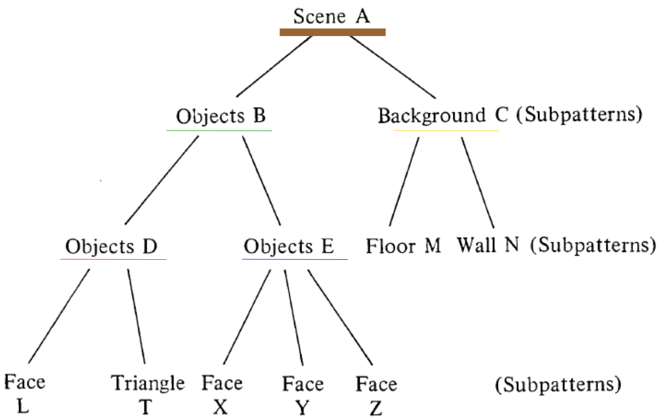
<Scene A> → <Objects B> <Background C>
<Objects B> → <Objects D> <Objects E>
<Objects D> → <Face L> <Triangle T>
<Objects E> → <Face X> <Face Y> <Face Z>
<Background C> → <Floor M> <Wall N>

(((LT)(XYZ))(MN))

Gramáticas

- Uma gramática é capaz de representar um **conjunto de sequências** (cadeias)
- → representa um padrão
- Ex2: sequências que representam imagens que seguem um padrão





<Scene A> → <Objects B> <Background C>
<Objects B> → <Objects D> <Objects E>
<Objects D> → <Face L> <Triangle T>
<Objects E> → <Face X> <Face Y> <Face Z>
<Background C> → <Floor M> <Wall N>

(((LT)(XYZ))(MN))

Gramáticas

- Uma gramática é capaz de representar um **conjunto de sequências** (cadeias)
- → representa um padrão
- A linguagem pode ser infinita (ex: todas as sequências de dígitos que começam com 1), mas pode ser representada por uma gramática finita (conjunto finito de elementos)
- Pode ser definida pelo especialista
- Pode ser aprendida (inferência gramatical)
- Pode enumerar (gerar) sequências desse padrão
- Pode ser utilizada para analisar se uma dada sequência pertence a esse padrão (classificação)

Agora a parte “chata”...

algumas definições formais

Operador * e variantes

- 0 ou mais concatenações de símbolos do conjunto ao qual o operador está sendo aplicado
- Ex: D = {0,1,2,3,4,5,6,7,8,9} e L = {a,b,c,…,z,A,B,C,…,Z}
  - D*: 0 ou mais dígitos
  - Um número segue o padrão DD* (ou pertence ao conjunto DD*}
  - Uma variável (de uma dada linguagem de programação) poderia ser definida como uma cadeia que segue o padrão L(D U L)*
- +: abreviação para 1 ou mais
- n: n concatenações
- Ex:
  - D^+ = DD*
  - D^nL^n: n dígitos seguidos de n letras (mesmo número de dígitos e letras, mas todos os dígitos vêm antes)

Linguagem e alfabeto

- Uma linguagem é um conjunto de cadeias sobre um conjunto Σ de símbolos (isto é, um subconjunto de Σ*)
- Σ é normalmente chamado de alfabeto (conjunto de símbolos que aparecem nas suas cadeias de interesse). Ex:
  - Σ = {A, C, G, T} para sequências de DNA
  - Σ = {0,1} para representar pixels de uma imagem em preto e branco
  - Σ = cada caracter digitável em um programa de computador
  - Σ = as várias palavras possíveis de um idioma
  - …
- ε ou λ : string vazia (corresponde a “”) - não pertence a Σ (mas pertence a Σ*)
- A linguagem é o que se quer representar (cada classe)

Gramáticas

- Definição: uma **gramática** G é uma quádrupla (V, Σ, S, P), na qual
  - V é o conjunto de símbolos não-terminais (ou variáveis)
  - Σ é o conjunto de símbolos terminais
  - S é o símbolo inicial
  - P é o conjunto de produções da forma

    $(\Sigma \cup V)^* V (\Sigma \cup V)^* \rightarrow (\Sigma \cup V)^*$

Gramáticas

- Uma **forma sentencial** de uma gramática G é qualquer cadeia obtida pela aplicação recorrente das seguintes regras:
  - S (símbolo inicial de G) é uma forma sentencial
  - Sejam $\alpha\rho\beta$ uma forma sentencial de G e $\rho \rightarrow \gamma$ uma produção de G. Então $\alpha\gamma\beta$ é também uma forma sentencial de G .

$(\alpha,\beta,\gamma \in (\Sigma \cup V)^*$ e $\rho \in (\Sigma \cup V)^* V (\Sigma \cup V)^*)$

Ex:
<Frase> → <sujeito> <predicado>

“<Frase>” é uma forma sentencial
Então “<sujeito> <predicado>” também é

Gramáticas

- Uma **forma sentencial** de uma gramática G é qualquer cadeia obtida pela aplicação recorrente das seguintes regras:
  - S (símbolo inicial de G) é uma forma sentencial
  - Sejam $\alpha\rho\beta$ uma forma sentencial de G e $\rho \rightarrow \gamma$ uma produção de G. Então $\alpha\gamma\beta$ é também uma forma sentencial de G .

  $(\alpha,\beta,\gamma \in (\Sigma \cup V)^*$ e $\rho \in (\Sigma \cup V)^* V (\Sigma \cup V)^*)$

- Derivação direta:
  - $\alpha\rho\beta \Rightarrow \alpha\gamma\beta$

Ex:
<predicado> → <verbo> <adjetivo>

a moça <predicado> => a moça <verbo> <adjetivo>

Gramáticas

- Derivação: aplicação de zero ou mais derivações diretas
  - α =>* μ
  - isto é, α => β => … => μ
- Uma cadeia w (w Є Σ*) é uma sentença de G se S =>* w (S sendo o símbolo inicial)
- Linguagem gerada por G:

  L(G) = { w Є Σ* | S =>* w }

Gramáticas - Exemplos

- $G = (V, \Sigma, S, P)$, onde
  - $V = \{S, A\}$
  - $\Sigma = \{0,1,2,3\}$
  - $S = S$
  - $P = \{$

    $S \rightarrow 0S33$

    $S \rightarrow A$

    $A \rightarrow 12$

    $A \rightarrow \varepsilon$

  $\}$

Gramáticas - Exemplos

- G = (V, Σ, S, P), onde
  - V = {S, A}
  - Σ = {0,1,2,3}
  - S = S
  - P = {

    $S \rightarrow 0S33$

    $S \rightarrow A$

    $A \rightarrow 12$

    $A \rightarrow \varepsilon$

  }

- Ex de formas sentenciais:

S, 0S33, 00S3333, 00A3333

- 0S33 => 00S3333
- 0S33 =>* 00A3333
- 0S33 =>* 0S33
- Ex de sentenças:

00123333, 12, ε

- L(G) =

Gramáticas - Exemplos

- $G = (V, \Sigma, S, P)$, onde
  - $V = \{S, A\}$
  - $\Sigma = \{0,1,2,3\}$
  - $S = S$
  - $P = \{$

    $S \rightarrow 0S33$

    $S \rightarrow A$

    $A \rightarrow 12$

    $A \rightarrow \varepsilon$

  $\}$

- Ex de formas sentenciais:

S, 0S33, 00S3333, 00A3333

- 0S33 => 00S3333
- 0S33 =>* 00A3333
- 0S33 =>* 0S33
- Ex de sentenças:

00123333, 12, ε

- $L(G) = \{0^m1^n2^n3^{2m} \mid m \geq 0 \text{ e } n = 0 \text{ ou } n=1\}$

Gramáticas - Simplificação

- $G = (V, \Sigma, S, P)$, onde
  - $V = \{S, A\}$
  - $\Sigma = \{0,1,2,3\}$
  - $S = S$
  - $P = \{$

    $S \rightarrow 0S33$

    $S \rightarrow A$

    $A \rightarrow 12$

    $A \rightarrow \varepsilon$

  $\}$

- $G = (V, \Sigma, S, P)$, onde
  - $V = \{S, A\}$
  - $\Sigma = \{0,1,2,3\}$
  - $S = S$
  - $P = \{$

    $S \rightarrow 0S33 \mid A$

    $A \rightarrow 12 \mid \varepsilon$

  $\}$

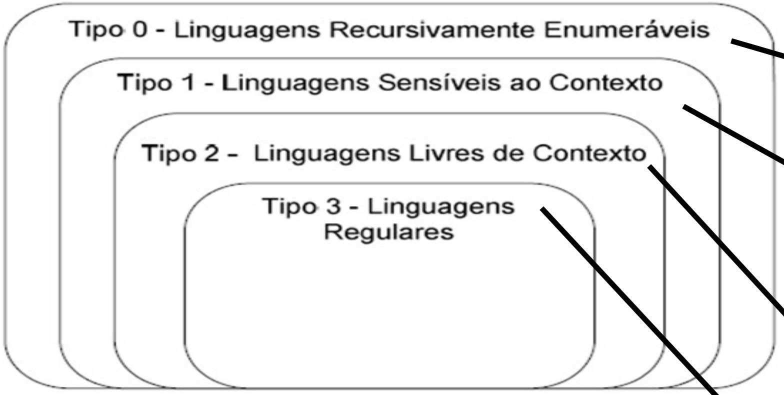
Gramáticas

- Gramáticas são dispositivos geradores (geram cadeias)
- Há dispositivos formais equivalentes (ex: autômatos, máquinas de Turing) que são reconhecedores (reconhecem se uma cadeia pertence à linguagem)
- Dada uma cadeia w, reconhecer se w Є L(G) é um processo chamado análise sintática
- Dependendo do formato das produções, a análise sintática pode ser mais ou menos complexa

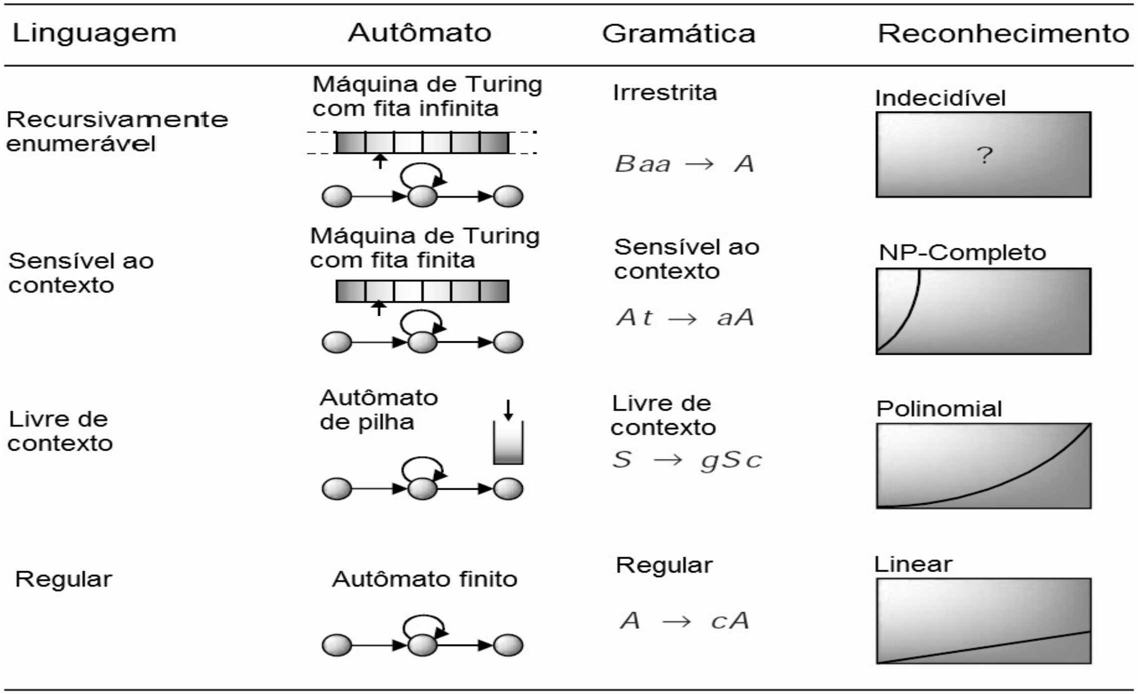
Hierarquia de Chomsky

- Hierarquia das linguagens em classes de acordo com a sua complexidade relativa (Noam Chomsky, 1956)
- Cada classe de linguagem pode ser gerada por um tipo de gramática (formato das produções)
- Cada tipo de gramática tem uma complexidade de análise sintática diferente
- Importância na prática: dada uma linguagem, saber qual o dispositivo mais eficiente para análise sintática

Hierarquia de Chomsky



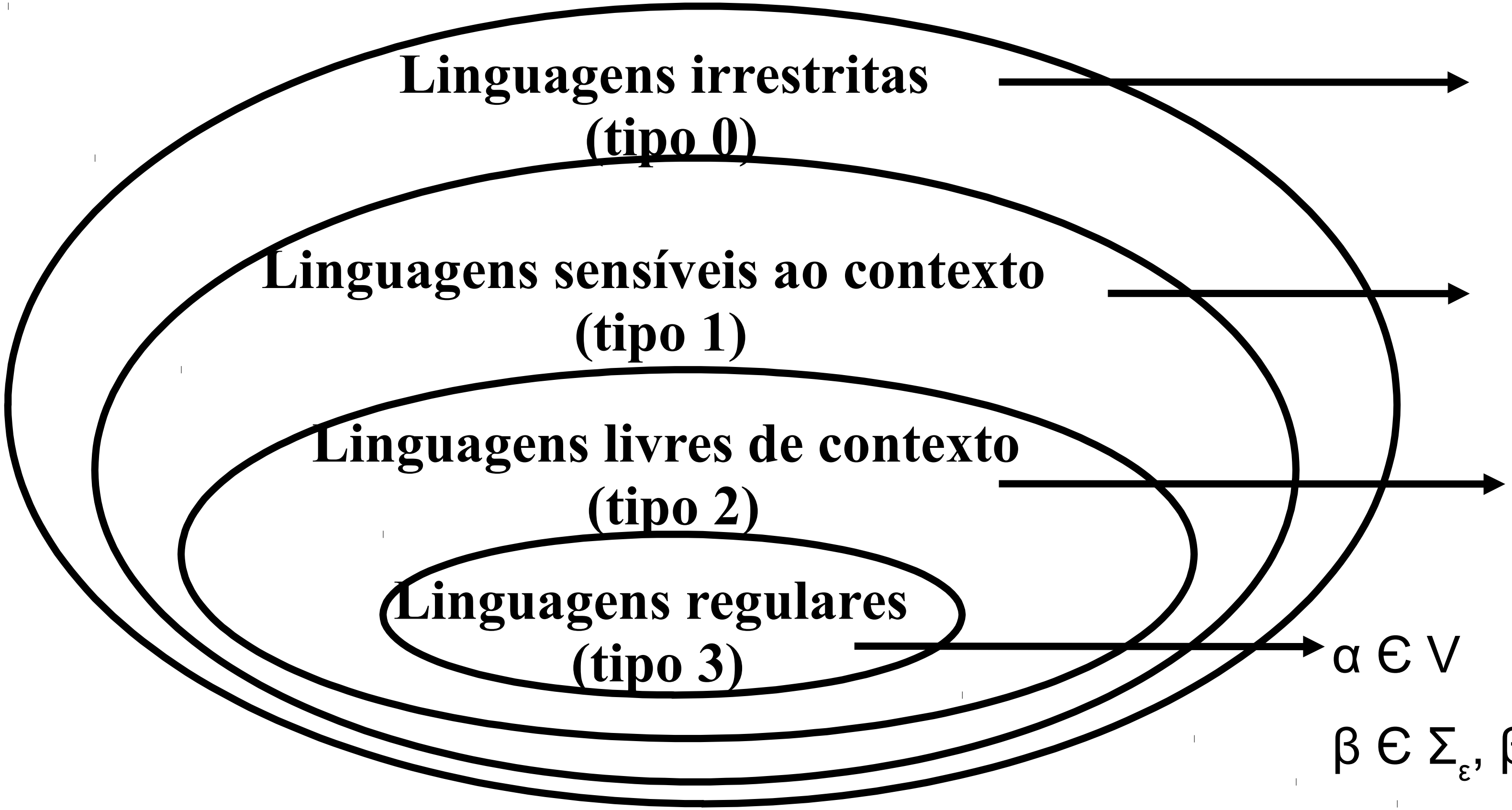
Fonte: Adaptado de Matsuno (2006)



Fonte: Adaptado de Searls (2002)

Hierarquia de Chomsky

$\alpha \rightarrow \beta$



$\alpha \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$
$\beta \in (V \cup \Sigma)^*$

$\alpha \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$
$\beta \in (V \cup \Sigma)^*$
$|\alpha| <= |\beta|$

$\alpha \in V$

$\beta \in (V \cup \Sigma)^*$

$\alpha \in V$

$\beta \in \Sigma_{\varepsilon}$, $\beta \in V$, $\beta \in$ ($V\Sigma$ou ΣV)

Fim do vídeo 1

Gramáticas - Conceitos Básicos e Hierarquia de Chomsky

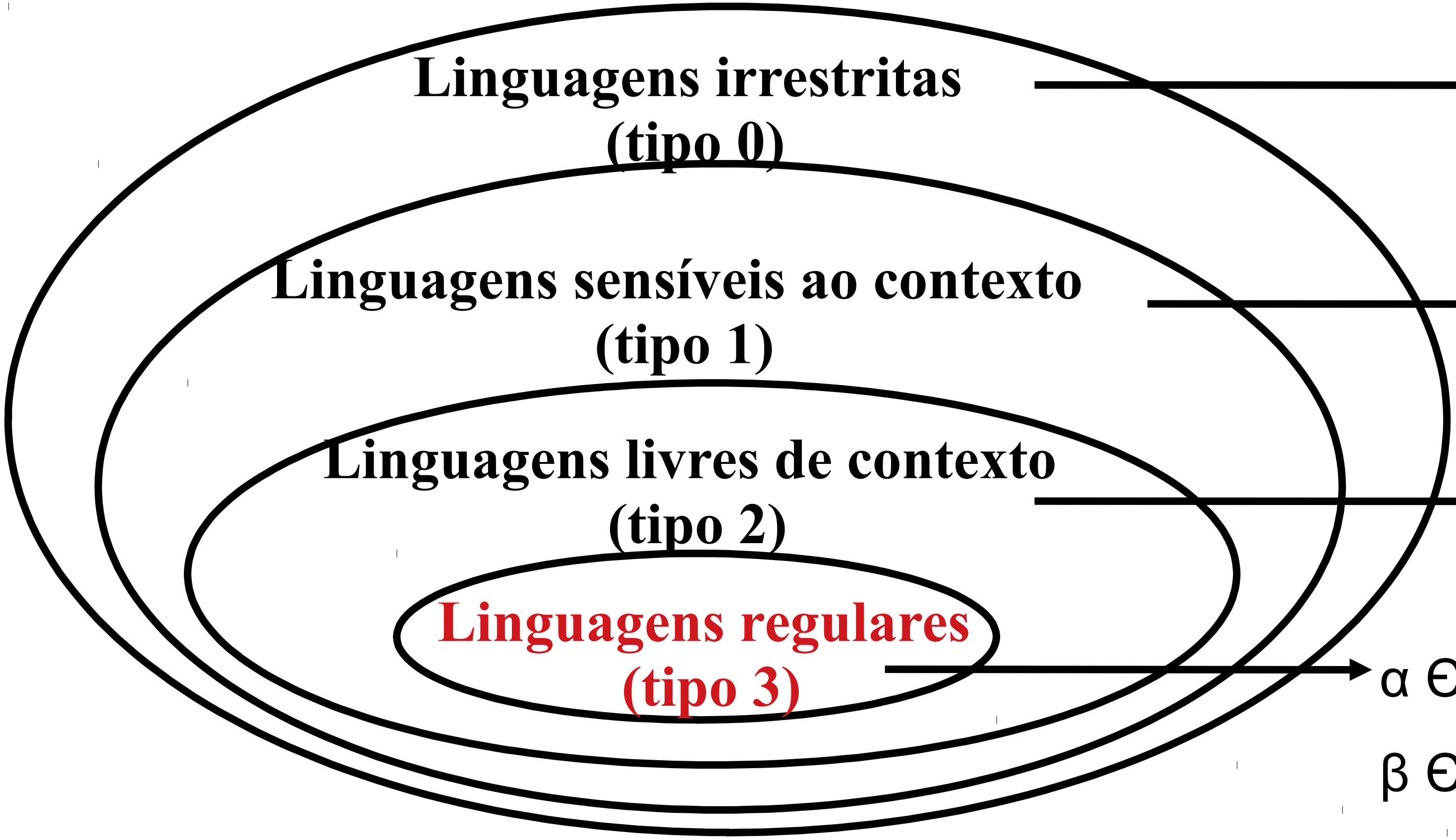
Professora:
Ariane Machado Lima

Vídeo 2

Gramáticas regulares

Professora:
Ariane Machado Lima

Hierarquia de Chomsky



$\alpha \rightarrow \beta$

$\alpha \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$
$\beta \in (V \cup \Sigma)^*$

$\alpha \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$
$\beta \in (V \cup \Sigma)^*$
$|\alpha| <= |\beta|$

$\alpha \in V$

$\beta \in (V \cup \Sigma)^*$

$\alpha \in V$

$\beta \in \Sigma_\varepsilon$, $\beta \in V$, $\beta \in (V\Sigma$ ou $\Sigma V)$

Gramáticas regulares

Uma gramática é **regular** se ela for

linear à esquerda ou **linear à direita**

Gramáticas lineares

$\alpha \rightarrow \beta$

- Gramática linear à esquerda:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in V\Sigma$

- Gramática linear à direita:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in \Sigma V$

Gramáticas lineares

$\alpha \rightarrow \beta$

- Gramática linear à esquerda:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in V\Sigma$

- Gramática linear à direita:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in \Sigma V$

Exemplos: para ambas, considere:
$V = \{S\}$
$\Sigma = \{0,1\}$
$S = S$

Linear à esquerda:

$P = \{ S \rightarrow S0,\ S \rightarrow S1,\ S \rightarrow 0,\ S \rightarrow 1\}$
Derivação da cadeia 1101
S

Gramáticas lineares

$\alpha \rightarrow \beta$

- Gramática linear à esquerda:
  - α Є V
  - β Є Σ ou β = ε ou β Є V ou β Є VΣ

- Gramática linear à direita:
  - α Є V
  - β Є Σ ou β = ε ou β Є V ou β Є ΣV

Exemplos: para ambas, considere:

V = {S}

Σ = {0,1}

S = S

Linear à esquerda:

P = { S → S0, S → S1, S → 0, S → 1}

Derivação da cadeia 1101

S => S1

Gramáticas lineares

$\alpha \rightarrow \beta$

- Gramática linear à esquerda:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in V\Sigma$
- Gramática linear à direita:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in \Sigma V$

Exemplos: para ambas, considere:

$V = \{S\}$

$\Sigma = \{0,1\}$

$S = S$

Linear à esquerda:

$P = \{ S \rightarrow S0, S \rightarrow S1, S \rightarrow 0, S \rightarrow 1\}$

Derivação da cadeia 1101

$S \Rightarrow S1 \Rightarrow S01$

Gramáticas lineares

$\alpha \rightarrow \beta$

- Gramática linear à esquerda:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in V\Sigma$
- Gramática linear à direita:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in \Sigma V$

Exemplos: para ambas, considere:

$V = \{S\}$

$\Sigma = \{0,1\}$

$S = S$

Linear à esquerda:

$P = \{ S \rightarrow S0, S \rightarrow S1, S \rightarrow 0, S \rightarrow 1\}$

Derivação da cadeia 1101

$S \Rightarrow S1 \Rightarrow S01 \Rightarrow S101$

Gramáticas lineares

$\alpha \rightarrow \beta$

- Gramática linear à esquerda:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in V\Sigma$

- Gramática linear à direita:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in \Sigma V$

Exemplos: para ambas, considere:

$V = \{S\}$

$\Sigma = \{0,1\}$

$S = S$

Linear à esquerda:

$P = \{ S \rightarrow S0,\ S \rightarrow S1,\ S \rightarrow 0,\ S \rightarrow 1\}$

Derivação da cadeia 1101

$S \Rightarrow S1 \Rightarrow S01 \Rightarrow S101 \Rightarrow 1101$

Gramáticas lineares

$\alpha \rightarrow \beta$

- Gramática linear à esquerda:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in V\Sigma$

- Gramática linear à direita:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in \Sigma V$

Exemplos: para ambas, considere:

$V = \{S\}$

$\Sigma = \{0,1\}$

$S = S$

Linear à esquerda:

$P = \{ S \rightarrow S0, S \rightarrow S1, S \rightarrow 0, S \rightarrow 1\}$

Derivação da cadeia 1101

$S \Rightarrow S1 \Rightarrow S01 \Rightarrow S101 \Rightarrow 1101$

Linear à direita:

$P = \{ S \rightarrow 0S, S \rightarrow 1S, S \rightarrow 0, S \rightarrow 1\}$

Derivação da cadeia 1101

S

Gramáticas lineares

$\alpha \rightarrow \beta$

- Gramática linear à esquerda:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in V\Sigma$
- Gramática linear à direita:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in \Sigma V$

Exemplos: para ambas, considere:
V = {S}
Σ = {0,1}
S = S

Linear à esquerda:

P = { S → S0, S → S1, S → 0, S → 1}
Derivação da cadeia 1101
S => S1 => S01 => S101 => 1101

Linear à direita:

P = { S → 0S, S → 1S, S → 0, S → 1}
Derivação da cadeia 1101
S => 1S

Gramáticas lineares

$\alpha \rightarrow \beta$

- Gramática linear à esquerda:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in V\Sigma$

- Gramática linear à direita:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in \Sigma V$

Exemplos: para ambas, considere:

$V = \{S\}$

$\Sigma = \{0,1\}$

$S = S$

Linear à esquerda:

$P = \{ S \rightarrow S0, S \rightarrow S1, S \rightarrow 0, S \rightarrow 1\}$

Derivação da cadeia 1101

S => S1 => S01 => S101 => 1101

Linear à direita:

$P = \{ S \rightarrow 0S, S \rightarrow 1S, S \rightarrow 0, S \rightarrow 1\}$

Derivação da cadeia 1101

S => 1S => 11S

Gramáticas lineares

$\alpha \rightarrow \beta$

- Gramática linear à esquerda:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in V\Sigma$
- Gramática linear à direita:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in \Sigma V$

Exemplos: para ambas, considere:

$V = \{S\}$

$\Sigma = \{0,1\}$

$S = S$

Linear à esquerda:

$P = \{ S \rightarrow S0, S \rightarrow S1, S \rightarrow 0, S \rightarrow 1\}$

Derivação da cadeia 1101

S => S1 => S01 => S101 => 1101

Linear à direita:

$P = \{ S \rightarrow 0S, S \rightarrow 1S, S \rightarrow 0, S \rightarrow 1\}$

Derivação da cadeia 1101

S => 1S => 11S => 110S

Gramáticas lineares

$\alpha \rightarrow \beta$

- Gramática linear à esquerda:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in V\Sigma$
- Gramática linear à direita:
  - $\alpha \in V$
  - $\beta \in \Sigma$ ou $\beta = \varepsilon$ ou $\beta \in V$ ou $\beta \in \Sigma V$

Exemplos: para ambas, considere:
V = {S}
Σ = {0,1}
S = S

Linear à esquerda:

P = { $S \rightarrow S0$, $S \rightarrow S1$, $S \rightarrow 0$, $S \rightarrow 1$}
Derivação da cadeia 1101
S => S1 => S01 => S101 => 1101

Linear à direita:

P = { $S \rightarrow 0S$, $S \rightarrow 1S$, $S \rightarrow 0$, $S \rightarrow 1$}
Derivação da cadeia 1101
S => 1S => 11S => 110S => 1101

Gramáticas regulares

- Duas gramáticas são equivalentes se elas geram exatamente a mesma linguagem
- Toda gramática **linear à esquerda** é equivalente a uma gramática **linear à direita** e vice-versa (prova em [RAMOS, 2009]).
- Uma linguagem é regular se e somente se ela é gerada por uma gramática regular (ou seja, por uma gramática linear à esquerda ou à direita)

Exemplo

Vamos projetar uma gramática regular (linear à direita) para reconhecer essa linguagem:

```
atga
atgg
atta
aaga
cgag
```

$G = \{V, \Sigma, S, P\}$

$V = \{$

$\Sigma =$

$S =$

Exemplo

Vamos projetar uma gramática regular (linear à direita) para reconhecer essa linguagem:

```
atga
atgg
atta
aaga
cgag
```

$G = \{V, \Sigma, S, P\}$

$V = \{S_1,$

$\Sigma = \{a, c, g, t\}$

$S = S_1$

Exemplo

Vamos projetar uma gramática regular (linear à direita) para reconhecer essa linguagem:

```
atga
atgg
atta
aaga
cgag
```

$P = \{ S_1 \rightarrow$

$G = \{V, \Sigma, S, P\}$

$V = \{S_1,$

$\Sigma = \{a, c, g, t\}$

$S = S_1$

Exemplo

Vamos projetar uma gramática regular (linear à direita) para reconhecer essa linguagem:

```
atga
atgg
atta
aaga
cgag
```

$P = \{\ S_1 \rightarrow aS_2 \mid cS_3$

$G = \{V, \Sigma, S, P\}$

$V = \{S_1,$

$\Sigma = \{a, c, g, t\}$

$S = S_1$

Exemplo

Vamos projetar uma gramática regular (linear à direita) para reconhecer essa linguagem:

```
atga
atgg
atta
aaga
cgag
```

$G = \{V, \Sigma, S, P\}$

$V = \{S_1,$

$\Sigma = \{a, c, g, t\}$

$S = S_1$

$P = \{ S_1 \rightarrow aS_2 \mid cS_3$

$S_2 \rightarrow tS_4 \mid aS_5$

$S_4 \rightarrow gS_6 \mid tS_7$

$S_6 \rightarrow a \mid g$

Exemplo

Vamos projetar uma gramática regular (linear à direita) para reconhecer essa linguagem:

atga
atgg
atta
aaga
cgag

$G = \{V, \Sigma, S, P\}$

$V = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$

$\Sigma = \{a, c, g, t\}$

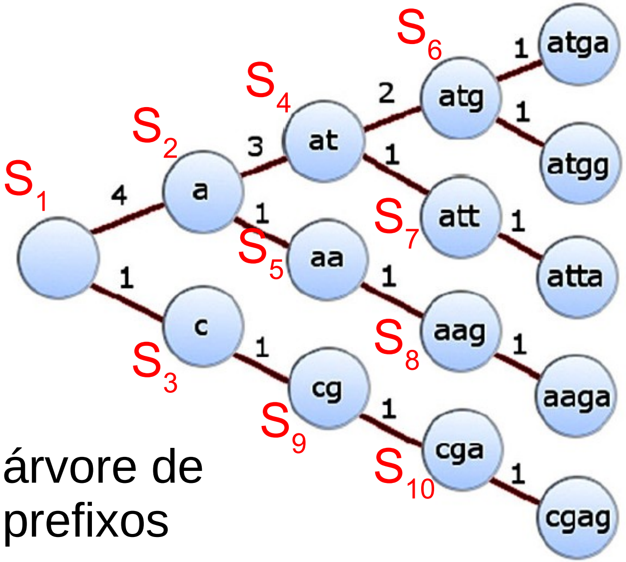
$S = S_1$

$P = \{ S_1 \rightarrow aS_2 \mid cS_3$
$S_2 \rightarrow tS_4 \mid aS_5$
$S_4 \rightarrow gS_6 \mid tS_7$
$S_6 \rightarrow a \mid g$
$S_7 \rightarrow a$
$S_5 \rightarrow gS_8$
$S_8 \rightarrow a$
$S_3 \rightarrow gS_9$
$S_9 \rightarrow aS_{10}$
$S_{10} \rightarrow g$
$\}$

Exemplo

Vamos projetar uma gramática regular (linear à direita) para reconhecer essa linguagem:

atga
atgg
atta
aaga
cgag



árvore de prefixos

$G = \{V, \Sigma, S, P\}$

$V = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$

$\Sigma = \{a, c, g, t\}$

$S = S_1$

$$
\begin{aligned}
P = \{\ & S_1 \rightarrow aS_2 \mid cS_3 \\
& S_2 \rightarrow tS_4 \mid aS_5 \\
& S_4 \rightarrow gS_6 \mid tS_7 \\
& S_6 \rightarrow a \mid g \\
& S_7 \rightarrow a \\
& S_5 \rightarrow gS_8 \\
& S_8 \rightarrow a \\
& S_3 \rightarrow gS_9 \\
& S_9 \rightarrow aS_{10} \\
& S_{10} \rightarrow g
\end{aligned}
$$

adaptado de:
Genetics and Molecular Research 6 (1): 105-115 (2007)

Exemplo

Vamos projetar uma gramática regular (linear à direita) para reconhecer essa linguagem:

atga
atgg

$G = \{V, \Sigma, S, P\}$

$P = \{ S_1 \rightarrow aS_2 \mid cS_3$

$S_2 \rightarrow tS_4 \mid aS_5$

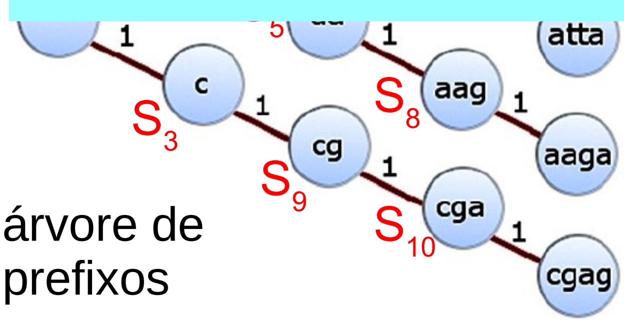
Conjuntos de sítios de ligação de fatores de transcrição são linguagens regulares

$S_3 \rightarrow gS_9$

$S_9 \rightarrow aS_{10}$

$S_{10} \rightarrow g$

$\}$



árvore de prefixos

Observação

E se eu projetar essa gramática?

```
atga
atgg
atta
aaga
cgag
```

$G = \{V, \Sigma, S, P\}$

$V = \{S_1, S_2, S_3, S_4\}$

$\Sigma = \{a, c, g, t\}$

$S = S_1$

$P = \{ S_1 \rightarrow aS_2 \mid cS_2$

$S_2 \rightarrow tS_3 \mid aS_3 \mid gS_3$

$S_3 \rightarrow tS_4 \mid aS_4 \mid gS_4$

$S_4 \rightarrow a \mid g$

$\}$

Observação

E se eu projetar essa gramática?

```
atga
atgg
atta
aaga
cgag
```

$G = \{V, \Sigma, S, P\}$

$V = \{S_1, S_2, S_3, S_4\}$

$\Sigma = \{a, c, g, t\}$

$S = S_1$

$$
\begin{aligned}
P = \{\ & S_1 \rightarrow aS_2 \mid cS_2 \\
& S_2 \rightarrow tS_3 \mid aS_3 \mid gS_3 \\
& S_3 \rightarrow tS_4 \mid aS_4 \mid gS_4 \\
& S_4 \rightarrow a \mid g \\
\}\ &
\end{aligned}
$$

A linguagem gerada por essa gramática incluirá outras sequências, por exemplo ctga

Fim do vídeo 2

Gramáticas regulares

Professora:
Ariane Machado Lima

Vídeo 3

Autômatos Finitos Determinísticos

Professora:
Ariane Machado Lima

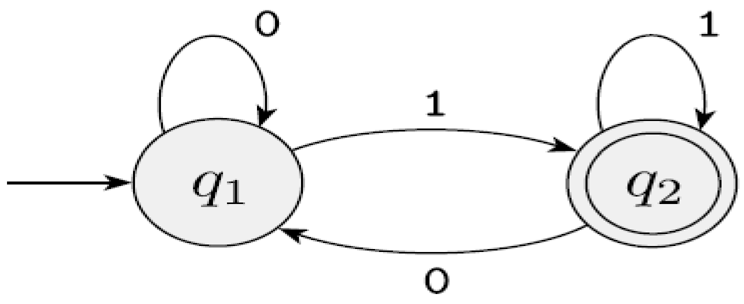
Gramáticas regulares e autômatos finitos

- Toda gramática **linear à esquerda** é equivalente a uma gramática **linear à direita** e vice-versa (prova em [RAMOS, 2009]).
- Toda gramática **linear à direita** é equivalente a um autômato finito, que é o dispositivo reconhecedor de linguagens regulares (prova mais à frente).
- Enquanto uma gramática regular gera cadeias de uma linguagem L, um autômato finito (para L) é capaz de analisar uma dada cadeia de entrada w e aceitá-la se w ϵ L ou rejeitá-la caso contrário (classificação).

Autômatos finitos

- Um autômato finito pode ser definido por um diagrama de estados

M1



Processo de reconhecimento:

Dados um autômato M e uma cadeia w:

- comece pelo estado inicial
- faça a transição de estados a cada símbolo lido (da cadeia de entrada w)
- ao finalizar a leitura, se M parou em um estado de aceitação aceite, e rejeite caso contrário

Círculos: estados

Círculos duplos: estados finais ou de aceitação

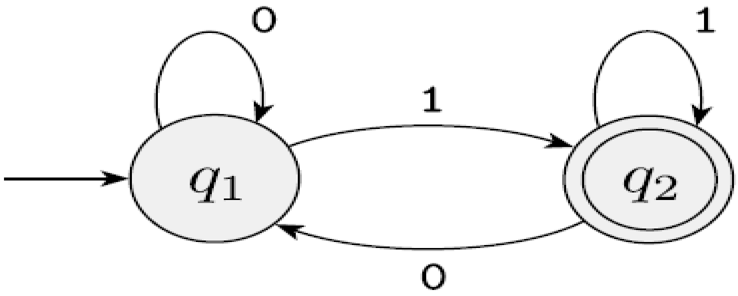
Seta sem início (somente uma): indica quem é o estado inicial

Setas entre estados: define a transição entre dois estados após a leitura do símbolo que está sobre a seta

Autômatos finitos

- Um autômato finito pode ser definido por um diagrama de estados

M1



Processo de reconhecimento:

Dados um autômato M e uma cadeia w:

- comece pelo estado inicial
- faça a transição de estados a cada símbolo lido (da cadeia de entrada w)
- ao finalizar a leitura, se M parou em um estado de aceitação aceite, e rejeite caso contrário

Exemplos de cadeias w aceitas pelo autômato M1:
1, 01, 11, 001, 00000001, 1101111, …

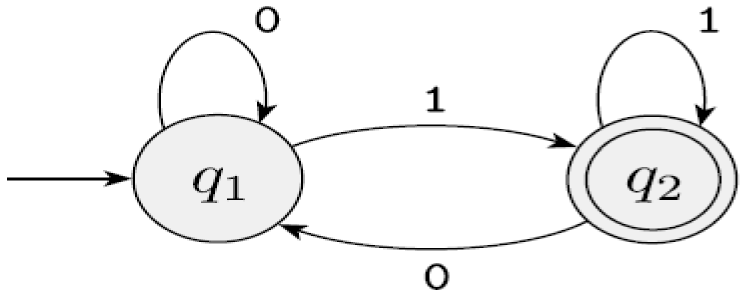
Exemplos de cadeias w NÃO aceitas pelo autômato M1:
0, 10, 110, 000, 0001010, 1111110, ε, ...

Cadeia vazia (“”)

Autômatos finitos

- Um autômato finito pode ser definido por um diagrama de estados

M1



Processo de reconhecimento:

Dados um autômato M e uma cadeia w:

- comece pelo estado inicial
- faça a transição de estados a cada símbolo lido (da cadeia de entrada w)
- ao finalizar a leitura, se M parou em um estado de aceitação aceite, e rejeite caso contrário

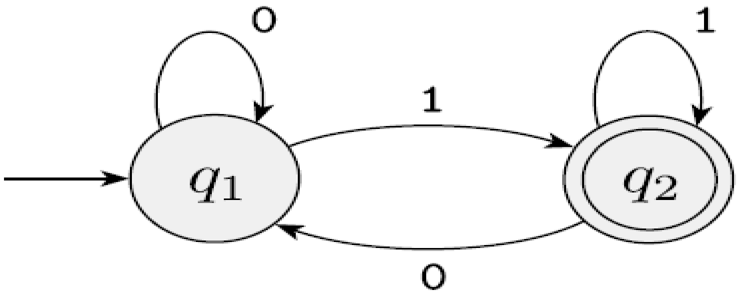
Que tipos de cadeias esse autômato M1 aceita?

Sequência binárias (de tamanho >= 1) que terminam em 1

Autômatos finitos

- Um autômato finito pode ser definido por um diagrama de estados

M1



Processo de reconhecimento:

Dados um autômato M e uma cadeia w:

- comece pelo estado inicial
- faça a transição de estados a cada símbolo lido (da cadeia de entrada w)
- ao finalizar a leitura, se M parou em um estado de aceitação aceite, e rejeite caso contrário

Que linguagem esse autômato M1 reconhece?

Sequência binárias (de tamanho >= 1) que terminam em 1

Exemplo

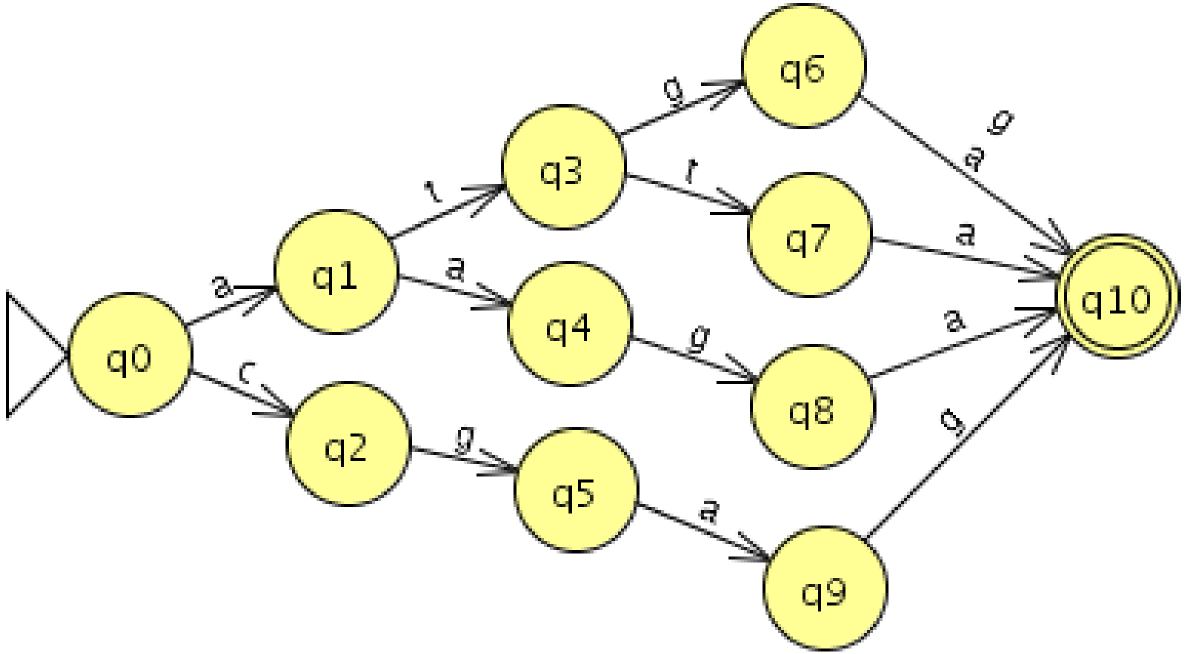
Vamos projetar um autômato para reconhecer essa linguagem:

```
atga
atgg
atta
aaga
cgag
```

Exemplo

Vamos projetar um autômato para reconhecer essa linguagem:

```
atga
atgg
atta
aaga
cgag
```



Definição formal de Autômatos Finitos

- Há dois tipos de autômatos finitos:
  - Determinísticos (AFD)
  - Não-determinísticos (AFN)

Autômatos Finitos **Determinísticos (AFD)**

- Definição formal:

Um ***autômato finito*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito conhecido como os ***estados***,
2. Σ é um conjunto finito chamado o ***alfabeto***,
3. $\delta : Q \times \Sigma \longrightarrow Q$ é a ***função de transição***,[1]
4. $q_0 \in Q$ é o ***estado inicial***, e
5. $F \subseteq Q$ é o ***conjunto de estados de aceitação***.[2]

Autômatos Finitos **Determinísticos (AFD)**

- Definição formal:

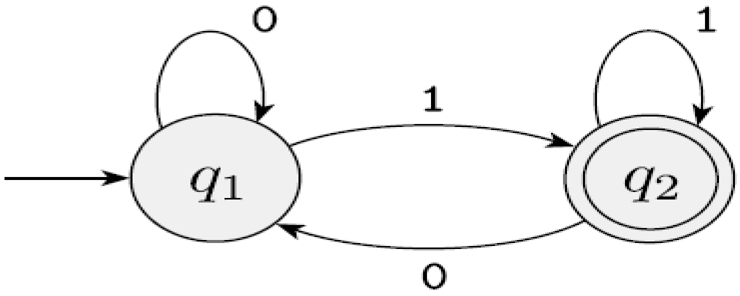
Um ***autômato finito*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito conhecido como os ***estados***,
2. Σ é um conjunto finito chamado o ***alfabeto***,
3. $\delta : Q \times \Sigma \longrightarrow Q$ é a ***função de transição***,[1]   total (definida para cada ponto do domínio)
4. $q_0 \in Q$ é o ***estado inicial***, e
5. $F \subseteq Q$ é o ***conjunto de estados de aceitação***.[2]

potencialmente vazio

Autômatos Finitos **Determinísticos (AFD)**

- Dado um estado atual e um símbolo de entrada sabemos exatamente para onde ir (está determinado)



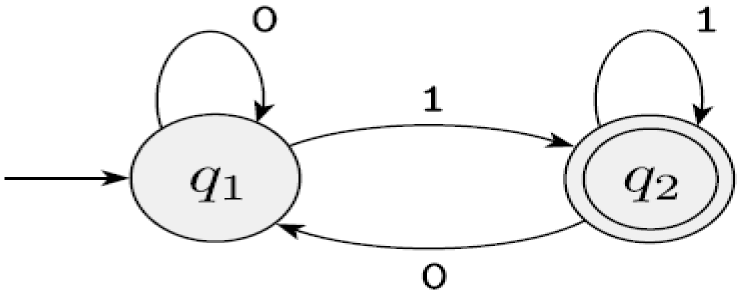
Um ***autômato finito*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito conhecido como os ***estados***,
2. Σ é um conjunto finito chamado o ***alfabeto***,
3. $\delta : Q \times \Sigma \longrightarrow Q$ é a ***função de transição***,[1]
4. $q_0 \in Q$ é o ***estado inicial***, e
5. $F \subseteq Q$ é o ***conjunto de estados de aceitação***.[2]

Para cada par (estado atual, próximo símbolo) está DETERMINADO qual é o próximo estado

Autômatos Finitos **Determinísticos (AFD)**

- Dado um estado atual e um símbolo de entrada sabemos exatamente para onde ir (está determinado)



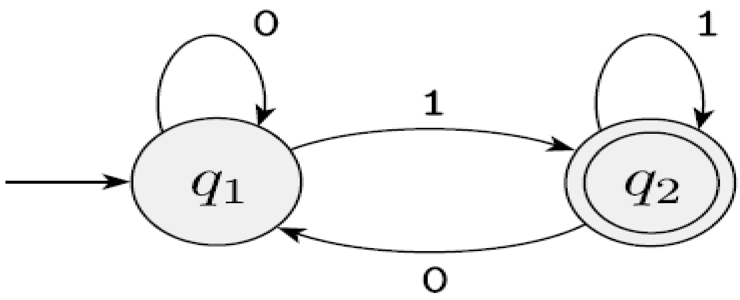
Um *autômato finito* é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito conhecido como os *estados*,
2. Σ é um conjunto finito chamado o *alfabeto*,
3. $\delta : Q \times \Sigma \longrightarrow Q$ é a *função de transição*,[1]
4. $q_0 \in Q$ é o *estado inicial*, e
5. $F \subseteq Q$ é o *conjunto de estados de aceitação*.[2]

Note que para um AFD deve haver, saindo de cada estado, uma aresta para CADA símbolo do alfabeto

Autômatos Finitos **Determinísticos (AFD)**

- Qual a definição formal do autômato M1?

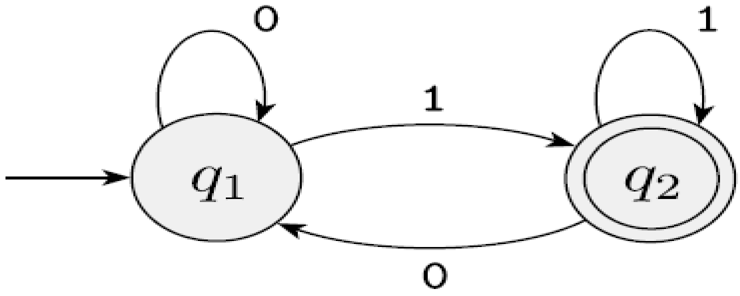


Um ***autômato finito*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito conhecido como os ***estados***,
2. Σ é um conjunto finito chamado o ***alfabeto***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ é a ***função de transição***,[1]
4. $q_0 \in Q$ é o ***estado inicial***, e
5. $F \subseteq Q$ é o ***conjunto de estados de aceitação***.[2]

Autômatos Finitos **Determinísticos (AFD)**

- Qual a definição formal do autômato M1?



Um ***autômato finito*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito conhecido como os ***estados***,
2. Σ é um conjunto finito chamado o ***alfabeto***,
3. $\delta \colon Q \times \Sigma \longrightarrow Q$ é a ***função de transição***,[1]
4. $q_0 \in Q$ é o ***estado inicial***, e
5. $F \subseteq Q$ é o ***conjunto de estados de aceitação***.[2]

Q = {q1, q2}

Σ = {0 ,1}

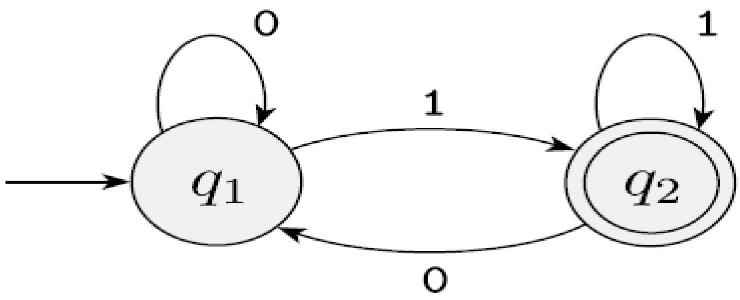
δ(q1, 0) = q1
δ(q1, 1) = q2
δ(q2, 0) = q1
δ(q2, 1) = q2

q0 = q1

F = {q2}

Autômatos Finitos **Determinísticos (AFD)**

- Qual a definição formal do autômato M1?



Um ***autômato finito*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

1. Q é um conjunto finito conhecido como os ***estados***,
2. Σ é um conjunto finito chamado o ***alfabeto***,
3. $\delta \colon Q \times \Sigma \longrightarrow Q$ é a ***função de transição***,[1]
4. $q_0 \in Q$ é o ***estado inicial***, e
5. $F \subseteq Q$ é o ***conjunto de estados de aceitação***.[2]

Q = {q1, q2}

Σ = {0 ,1}

δ(q1, 0) = q1
δ(q1, 1) = q2
δ(q2, 0) = q1
δ(q2, 1) = q2

Q \ Σ	**0**	**1**
q1	q1	q2
q2	q1	q2

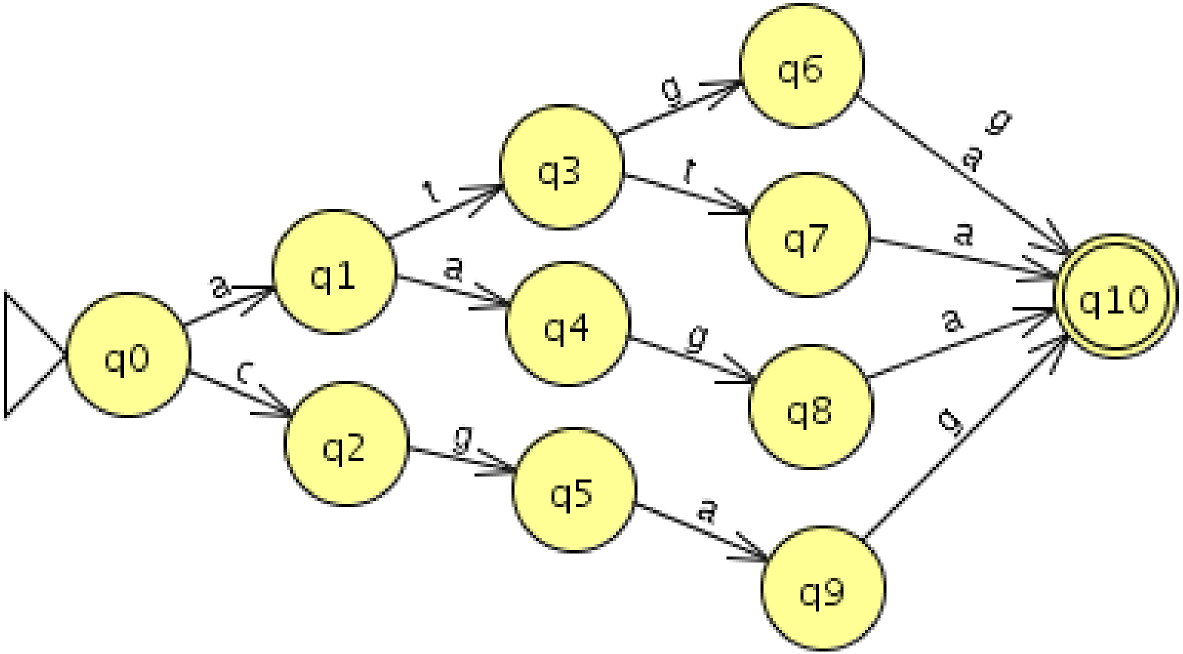
q0 = q1

F = {q2}

Exemplo

Este autômato é um AFD?

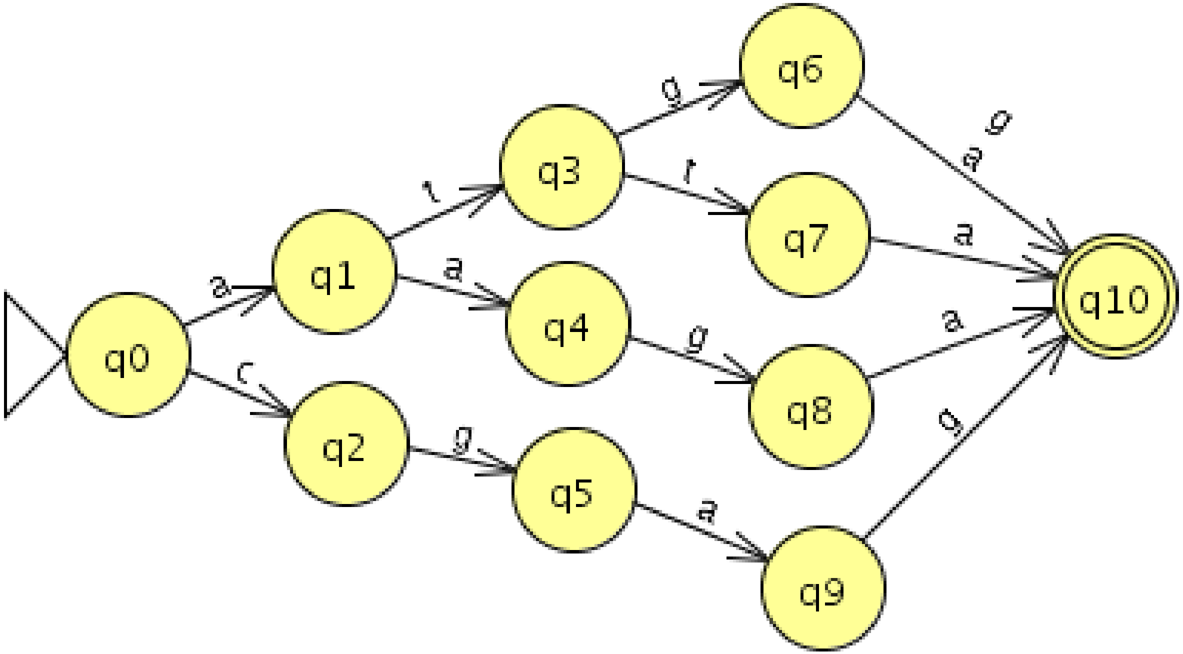
```
atga
atgg
atta
aaga
cgag
```



Exemplo

Este autômato é um AFD?

atga
atgg
atta
aaga
cgag

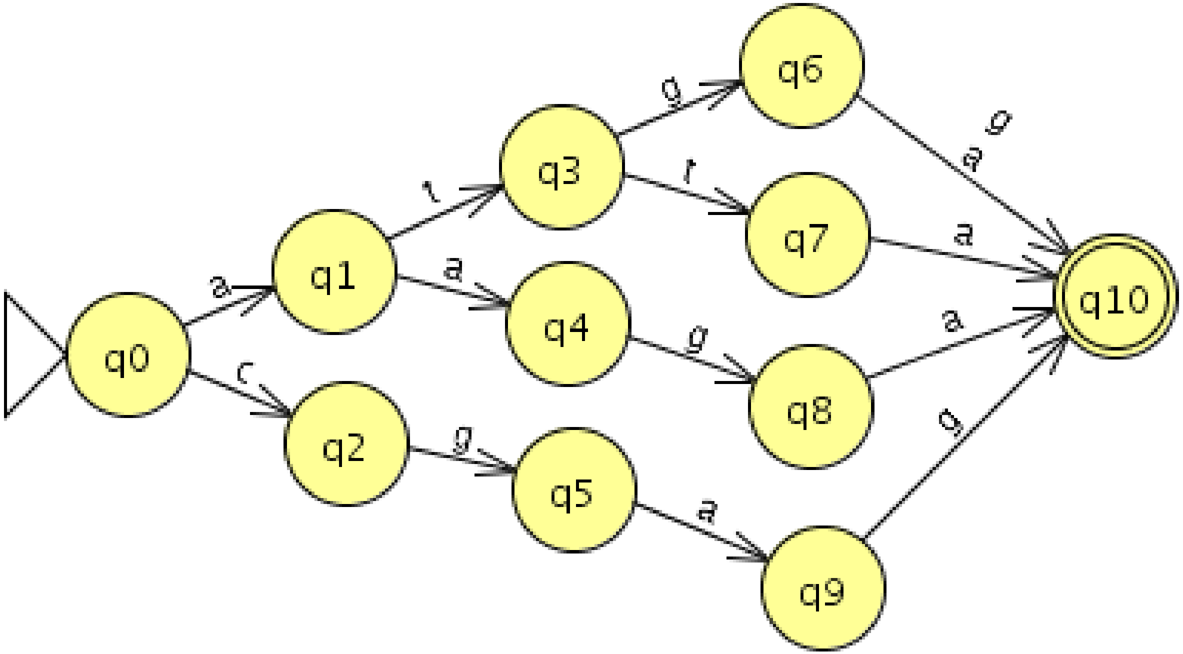


Q \ Σ	a	c	g	t
q0				
q1				
q2				
q3				
q4				
q5				
q6				
q7				
q8				
q9				
q10				

Exemplo

Este autômato é um AFD?

atga
atgg
atta
aaga
cgag

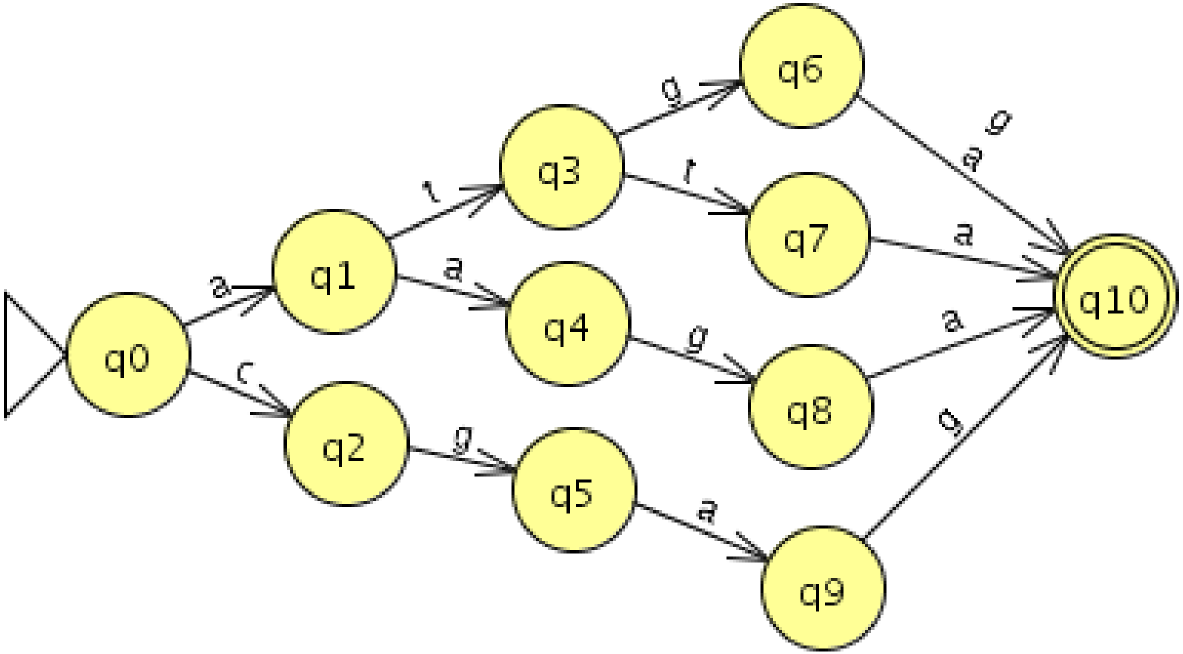


Q \ Σ	a	c	g	t
q0	q1	q2		
q1	q4			q3
q2			q5	
q3			q6	q7
q4			q8	
q5	q9			
q6	q10		q10	
q7	q10			
q8	q10			
q9			q10	
q10				

Exemplo

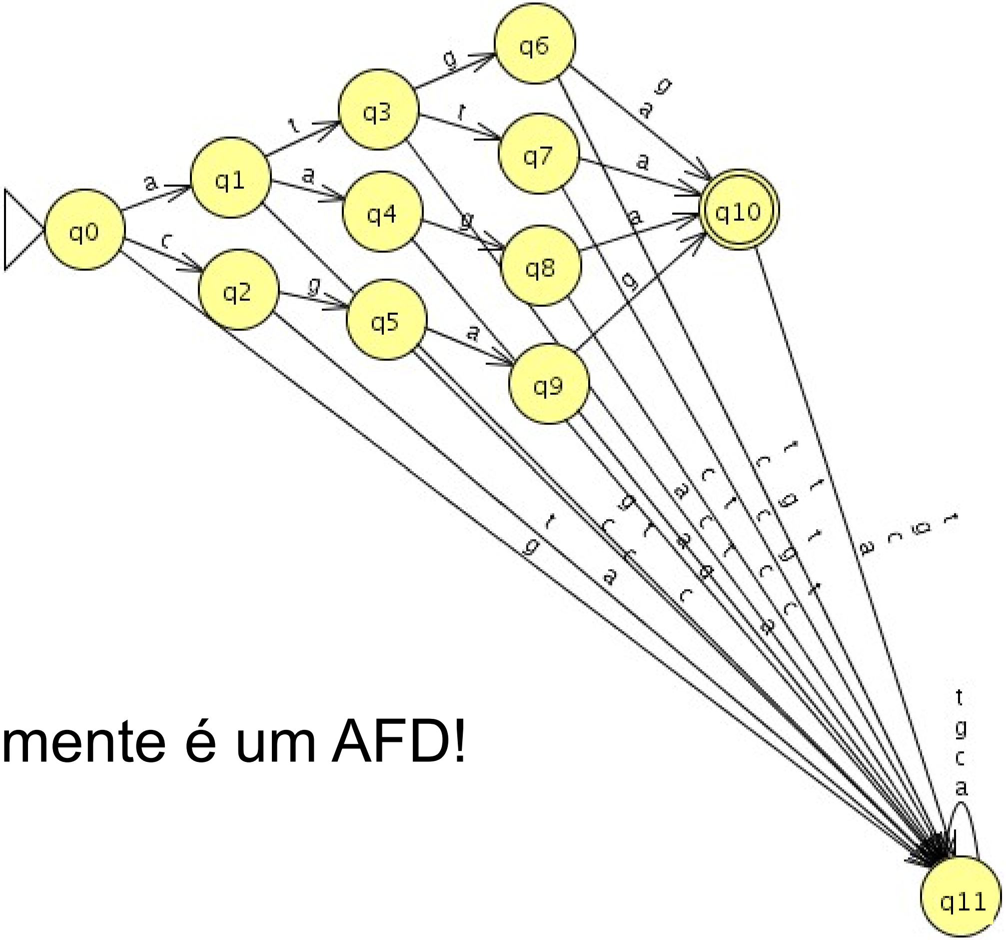
Este autômato é um AFD?

atga
atgg
atta
aaga
cgag



Q \ Σ	a	c	g	t
q0	q1	q2	q11	q11
q1	q4	q11	q11	q3
q2	q11	q11	q5	q11
q3	q11	q11	q6	q7
q4	q11	q11	q8	q11
q5	q9	q11	q11	q11
q6	q10	q11	q10	q11
q7	q10	q11	q11	q11
q8	q10	q11	q11	q11
q9	q11	q11	q10	q11
q10	q11	q11	q11	q11
q11	q11	q11	q11	q11

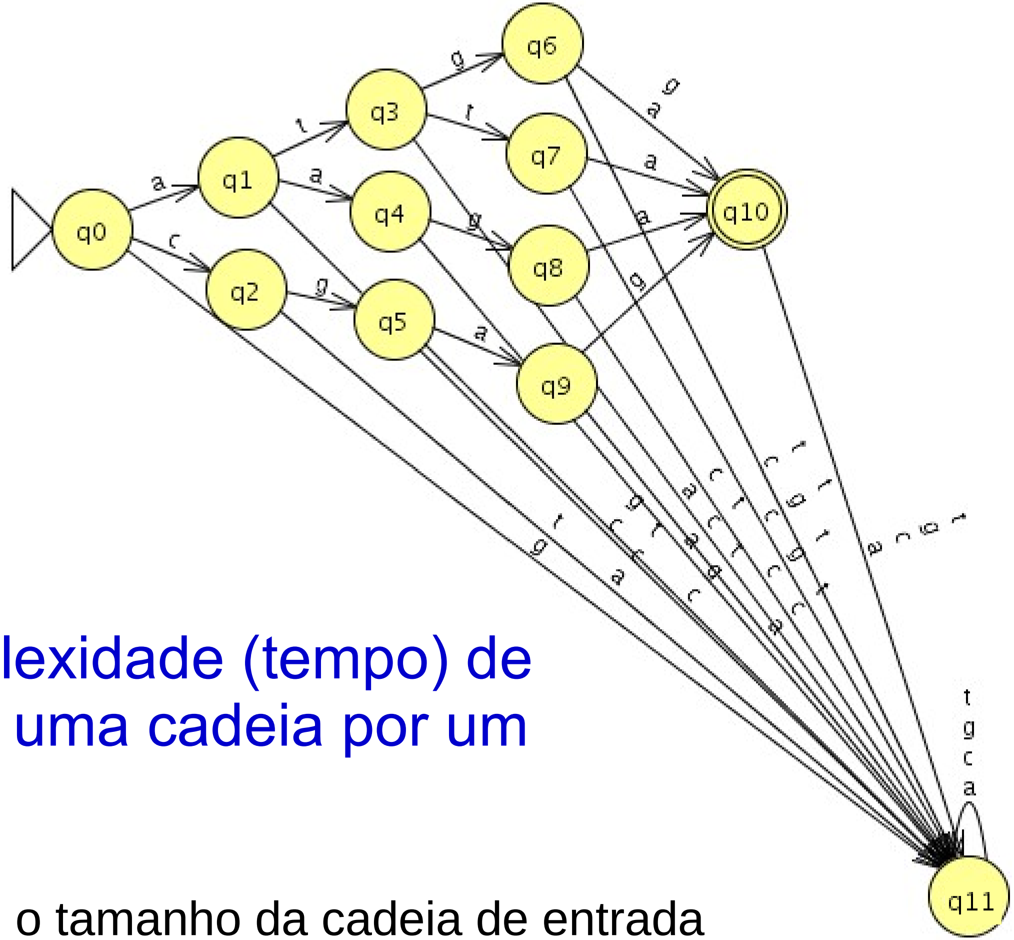
atga
atgg
atta
aaga
cgag



Agora certamente é um AFD!

Q \ Σ	a	c	g	t
q0	q1	q2	q11	q11
q1	q4	q11	q11	q3
q2	q11	q11	q5	q11
q3	q11	q11	q6	q7
q4	q11	q11	q8	q11
q5	q9	q11	q11	q11
q6	q10	q11	q10	q11
q7	q10	q11	q11	q11
q8	q10	q11	q11	q11
q9	q11	q11	q10	q11
q10	q11	q11	q11	q11
q11	q11	q11	q11	q11

atga
atgg
atta
aaga
cgag



Q \ Σ	a	c	g	t
q0	q1	q2	**q11**	**q11**
q1	q4	**q11**	**q11**	q3
q2	**q11**	**q11**	q5	**q11**
q3	**q11**	**q11**	q6	q7
q4	**q11**	**q11**	q8	**q11**
q5	q9	**q11**	**q11**	**q11**
q6	q10	**q11**	q10	**q11**
q7	q10	**q11**	**q11**	**q11**
q8	q10	**q11**	**q11**	**q11**
q9	**q11**	**q11**	q10	**q11**
q10	**q11**	**q11**	**q11**	**q11**
q11	**q11**	**q11**	**q11**	**q11**

Qual a complexidade (tempo) de análise de uma cadeia por um AFD?

O(n) – n sendo o tamanho da cadeia de entrada

Exercício

- Descreva o diagrama de estados de um AFD que aceita cadeias binárias (compostas por 0’s e 1’s) que comecem e terminem com zero, com tamanho pelo menos 1
    - 0, 00, 010, 000000, 0101110, ...

Fim do vídeo 3

Autômatos Finitos Determinísticos

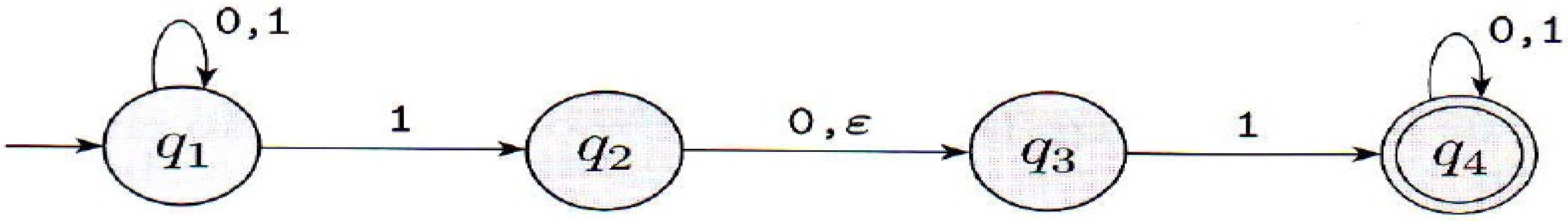
Professora:
Ariane Machado Lima

Vídeo 4

Autômatos Finitos Não-determinísticos

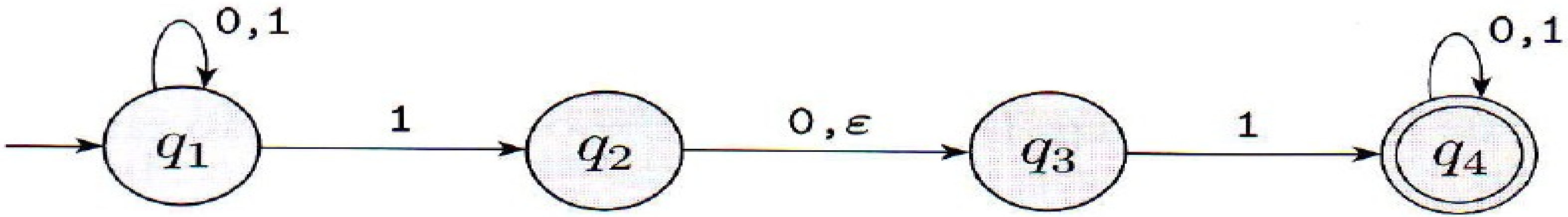
Professora:
Ariane Machado Lima

Autômatos Finitos
Não Determinísticos (AFN)



- Um estado pode ter 0 ou mais transições (setas saindo) para cada símbolo de Σ
- Um estado pode ter setas rotuladas por ε ou λ (string vazia)

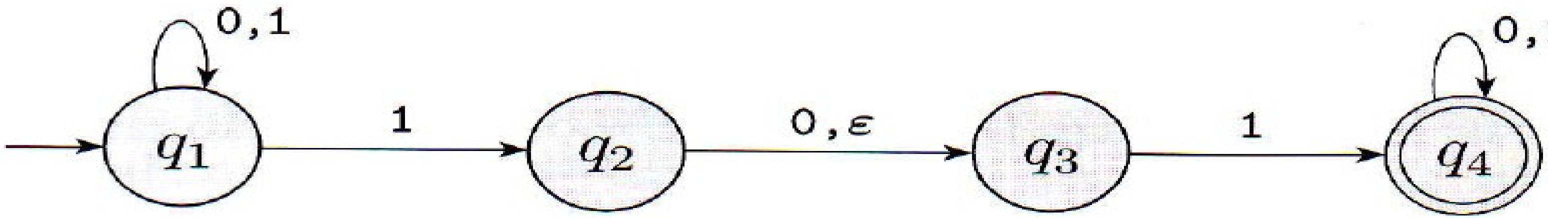
Autômatos Finitos
Não Determinísticos (AFN)



- Um estado pode ter 0 ou mais transições (setas saindo) para cada símbolo de Σ
- Um estado pode ter setas rotuladas por ε ou λ (string vazia)

Como ficaria a função de transição neste caso???

Autômatos Finitos
Não Determinísticos (AFN)



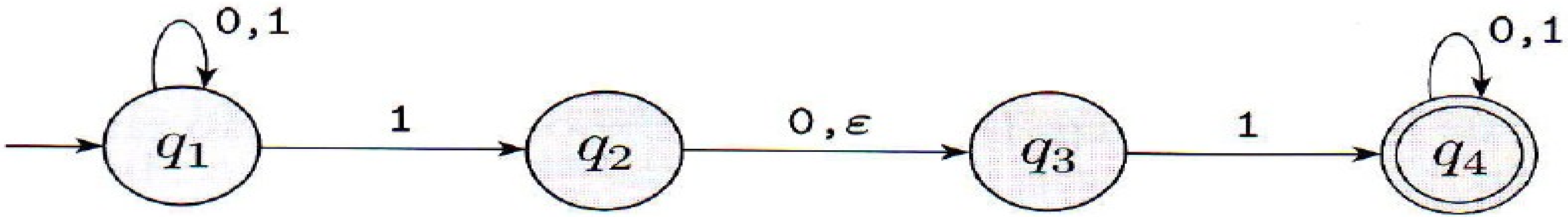
Um ***autômato finito não-determinístico*** é uma 5-upla $(Q, \Sigma, \delta, q_0, F)$, onde

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

1. Q é um conjunto finito de estados,
2. Σ é um alfabeto finito,
3. $\delta : Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ é a função de transição,
4. $q_0 \in Q$ é o estado inicial, e
5. $F \subseteq Q$ é o conjunto de estados de aceitação.

Conjunto potência

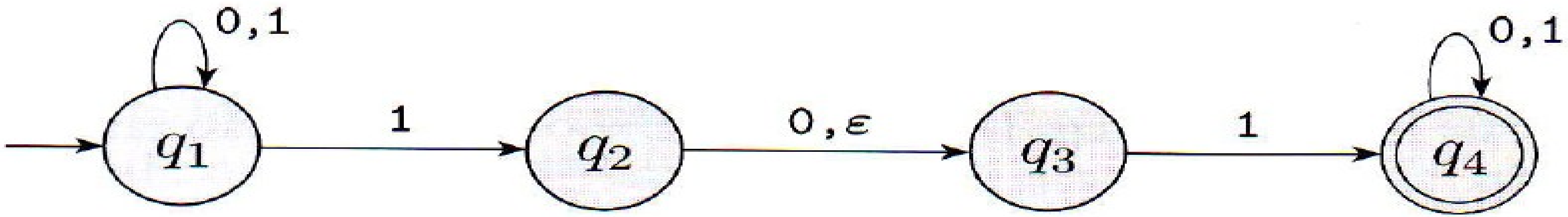
Autômatos Finitos Não Determinísticos (AFN)



- δ:

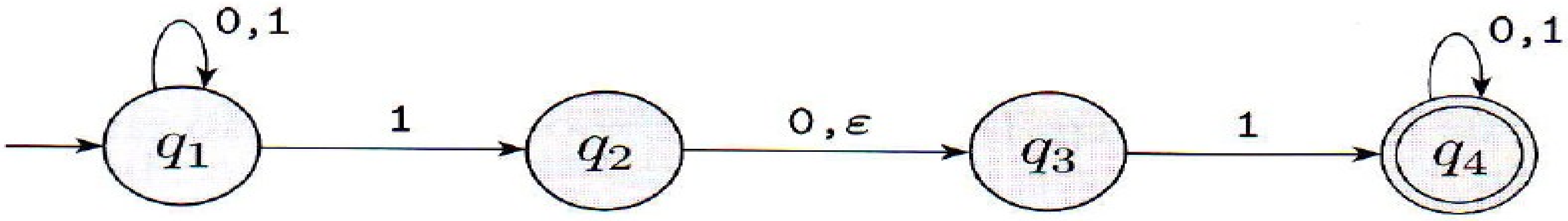
Q \ Σ	0	1	ε
q1	{q1}	{q1,q2)	∅
q2	{q3}	∅	{q3}
q3	∅	{q4}	∅
q4	{q4}	{q4}	∅

Autômatos Finitos Não Determinísticos (AFN)



- Que linguagem este AFN reconhece?

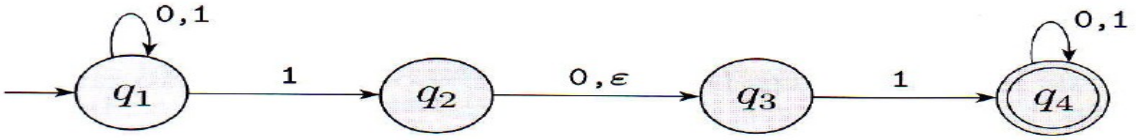
Autômatos Finitos Não Determinísticos (AFN)



- Que linguagem este AFN reconhece?
- Sequências binárias que contenham 11 ou 101

Funcionamento de um AFN

- Sempre que o autômato se depara com um não-determinismo (símbolo repetido ou ε) faz uma cópia de si (um clone), exatamente no ponto onde pausou, e cada cópia segue com uma alternativa, em paralelo, a partir daquele ponto.
- Se alguma cópia aceitar a cadeia, então o AFN aceita a cadeia
- As várias cópias são como várias *threads* ou processos executados em paralelo...



Árvore de computações

Símbolo lido

0

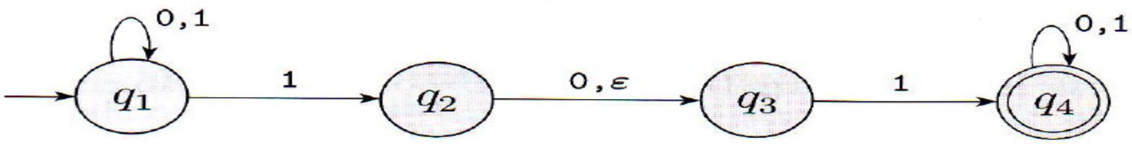
1

0

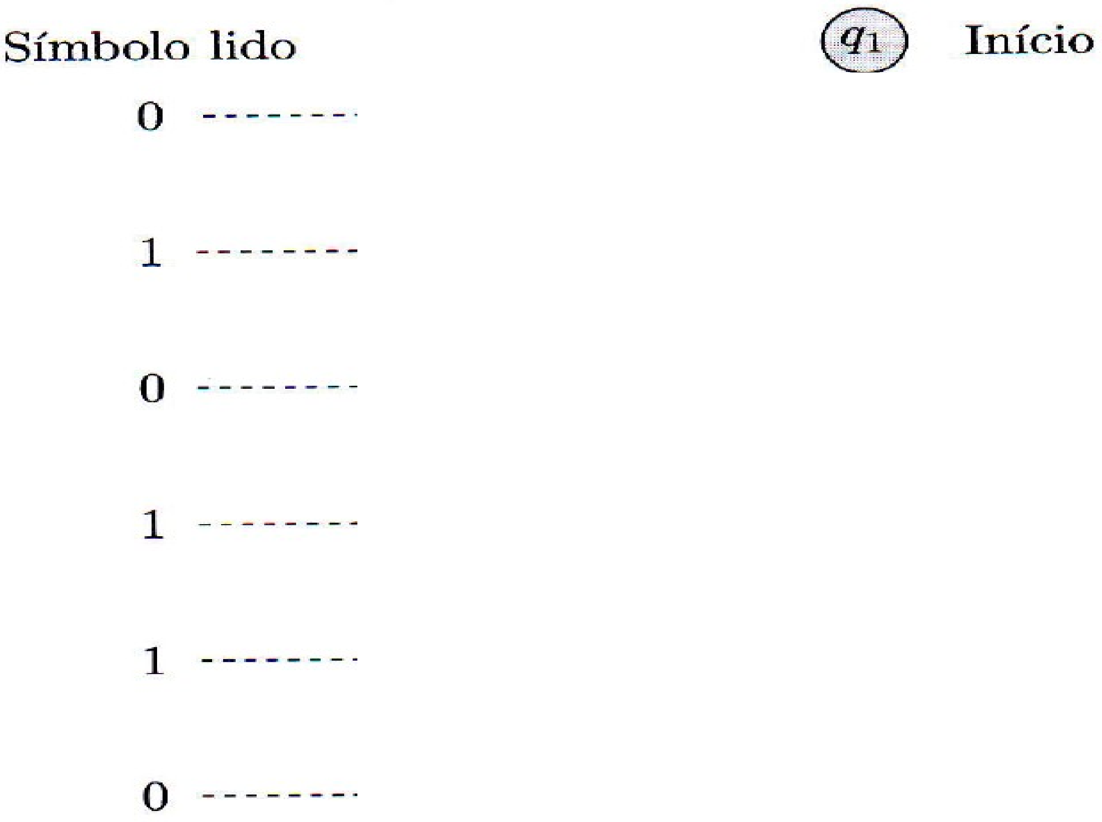
1

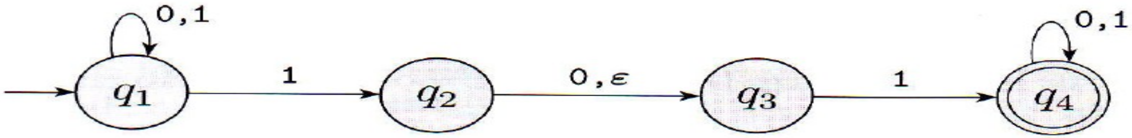
1

0

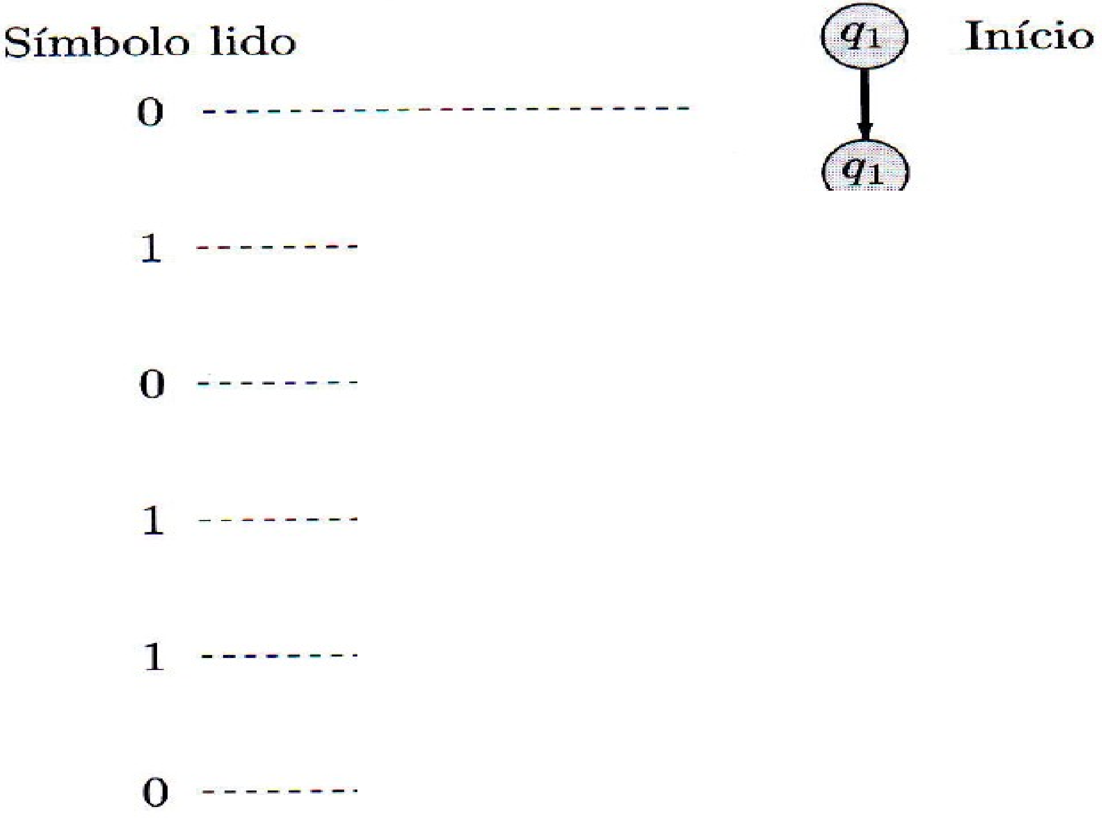


Árvore de computações

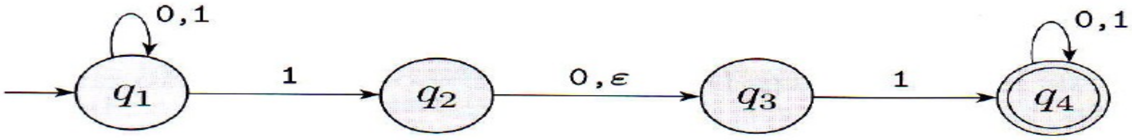


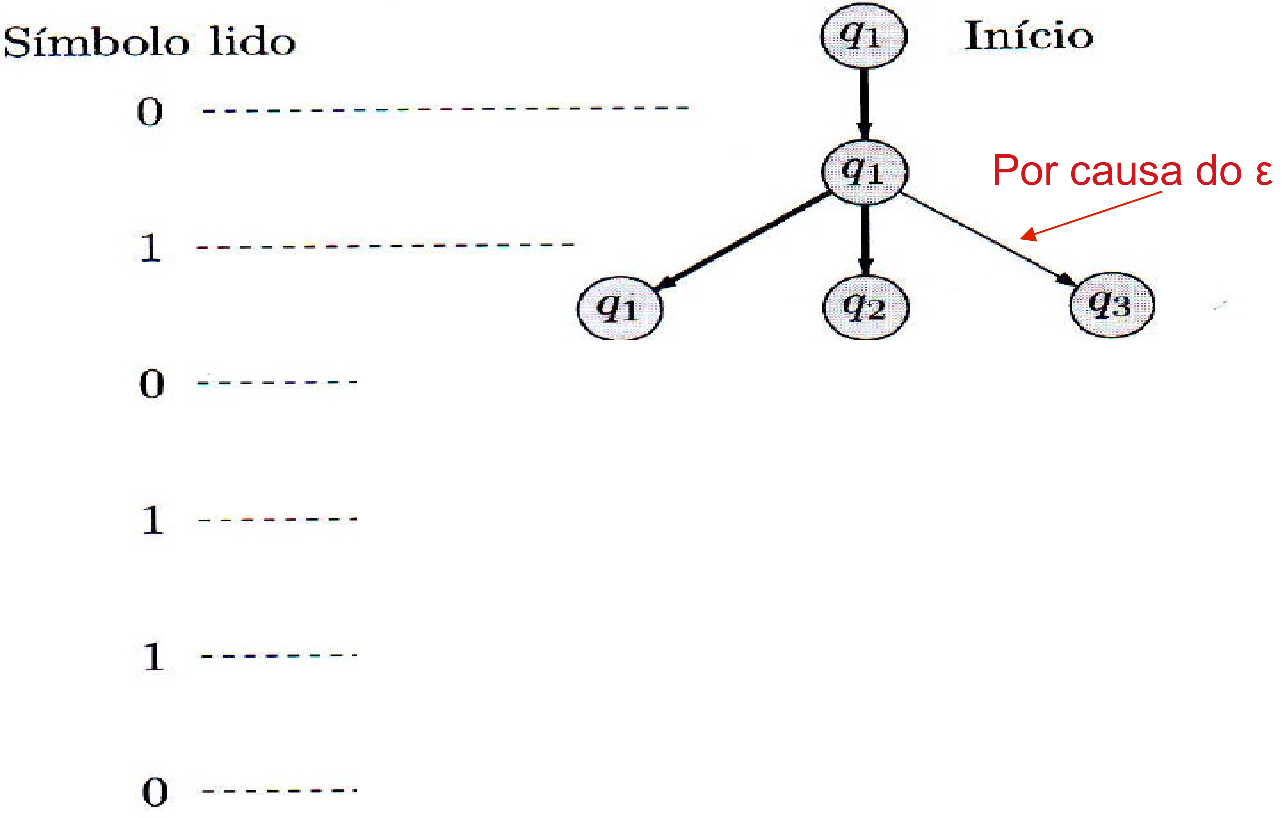


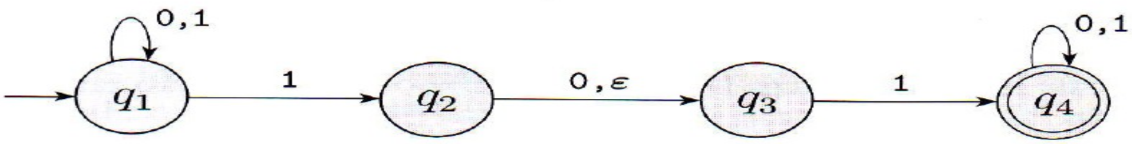
Árvore de computações



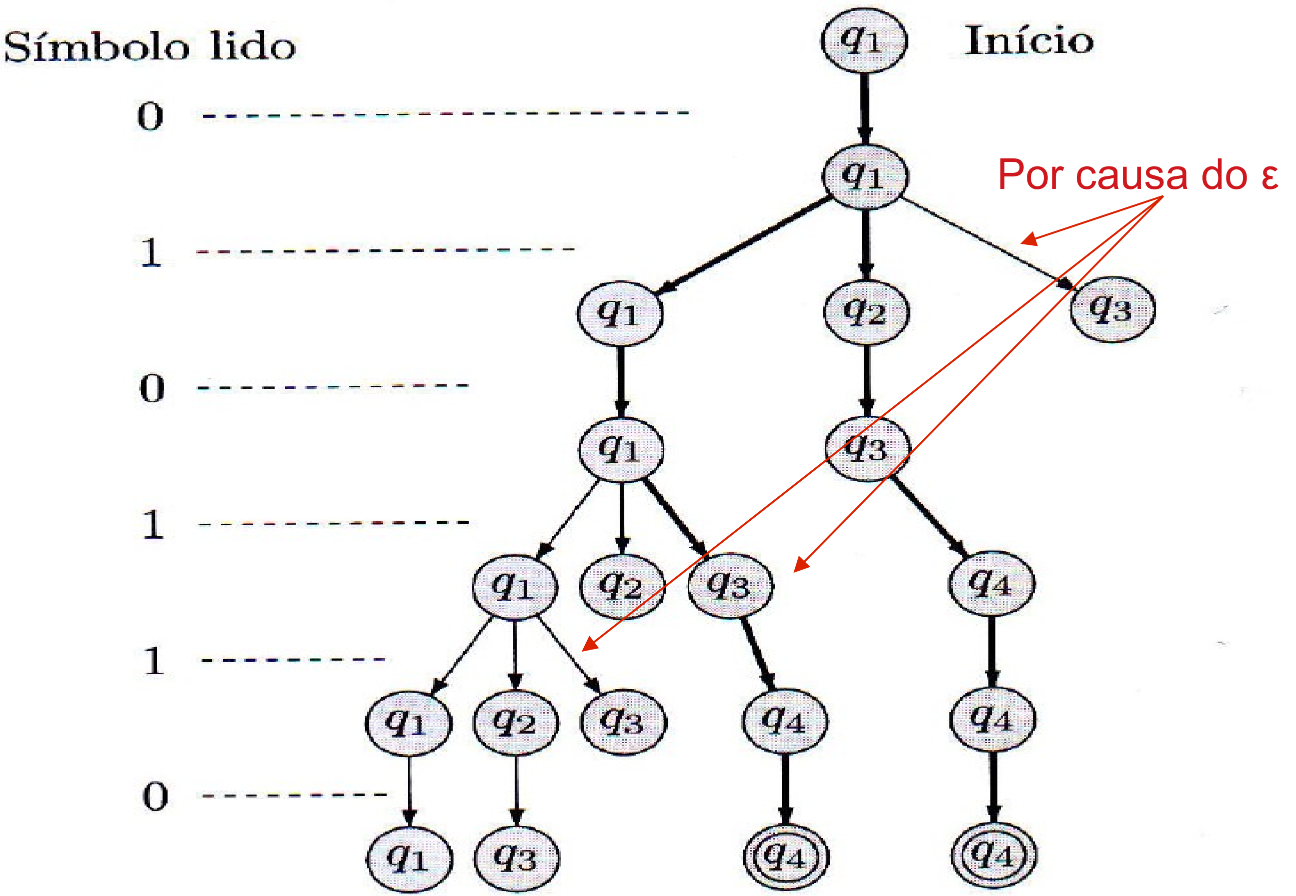
Árvore de computações

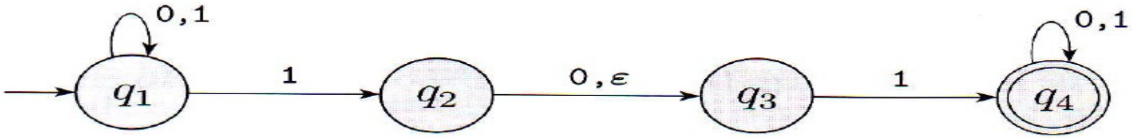




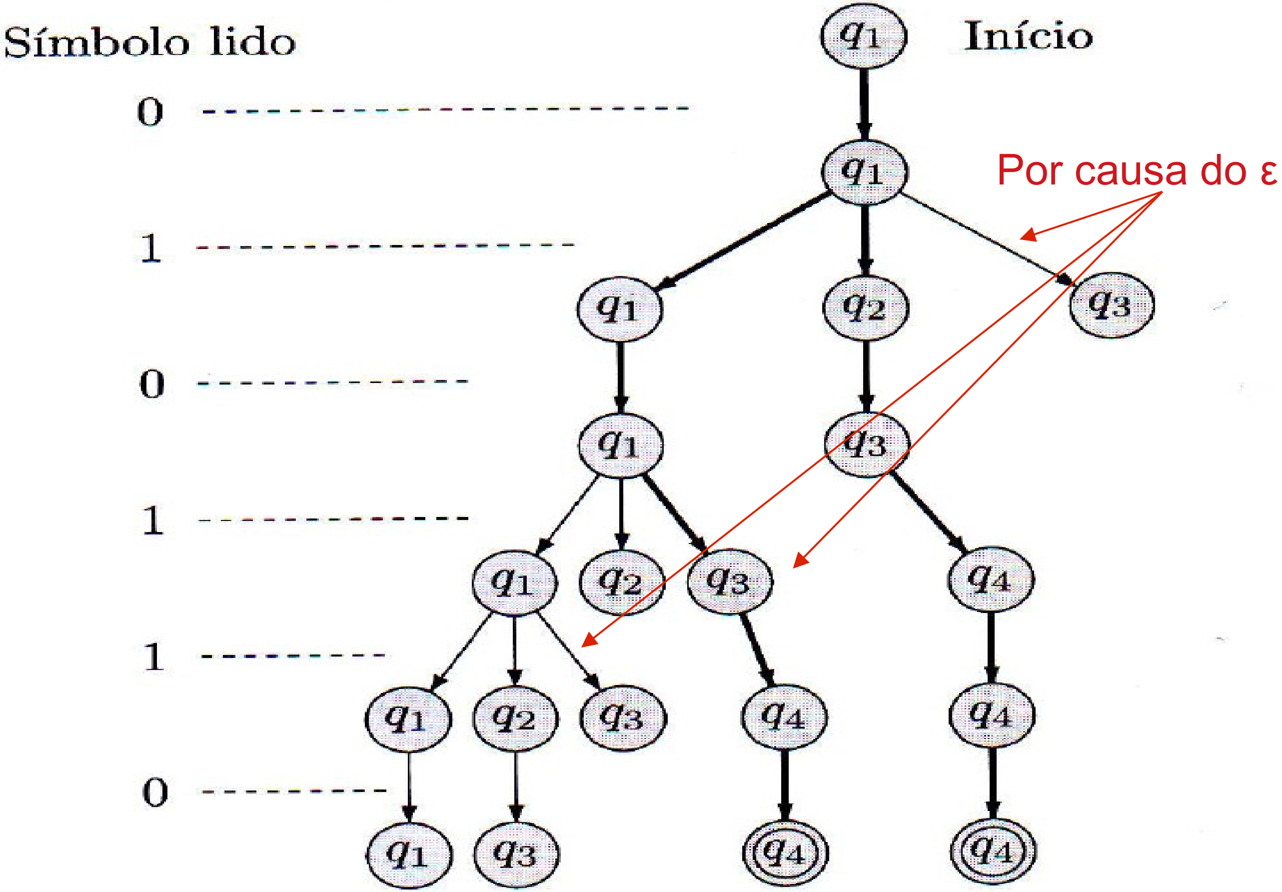


Árvore de computações





Dá para simular no JFlap



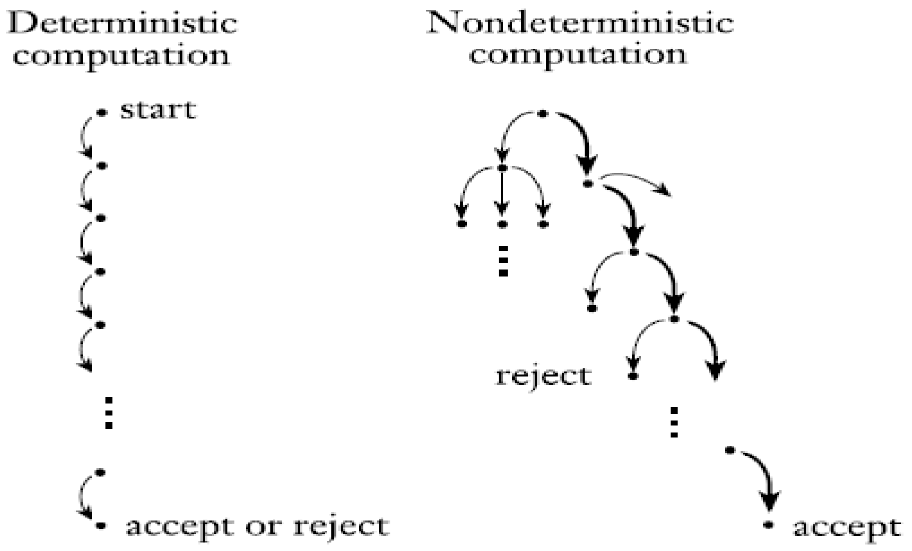


FIGURA 1.28
Computações determinísticas e não-determinísticas com um ramo de aceitação

AFD’s são mais eficientes que AFN’s (tempo)

Equivalência entre AFDs e AFNs

- Duas máquinas são equivalentes se elas reconhecem a mesma linguagem

TEOREMA 1.39 ....

Todo autômato finito não-determinístico tem um autômato finito determinístico equivalente.

AFDs e AFNs

- Por que o teorema de equivalência é importante?

AFDs e AFNs

- Por que o teorema de equivalência é importante?
- Pode-se optar por um ou outro dependendo do objetivo
- AFDs são mais eficientes
- AFNs podem:
  - ser mais fáceis de serem projetados
  - facilitar demonstração de teoremas
  - ser úteis em versões probabilísticas

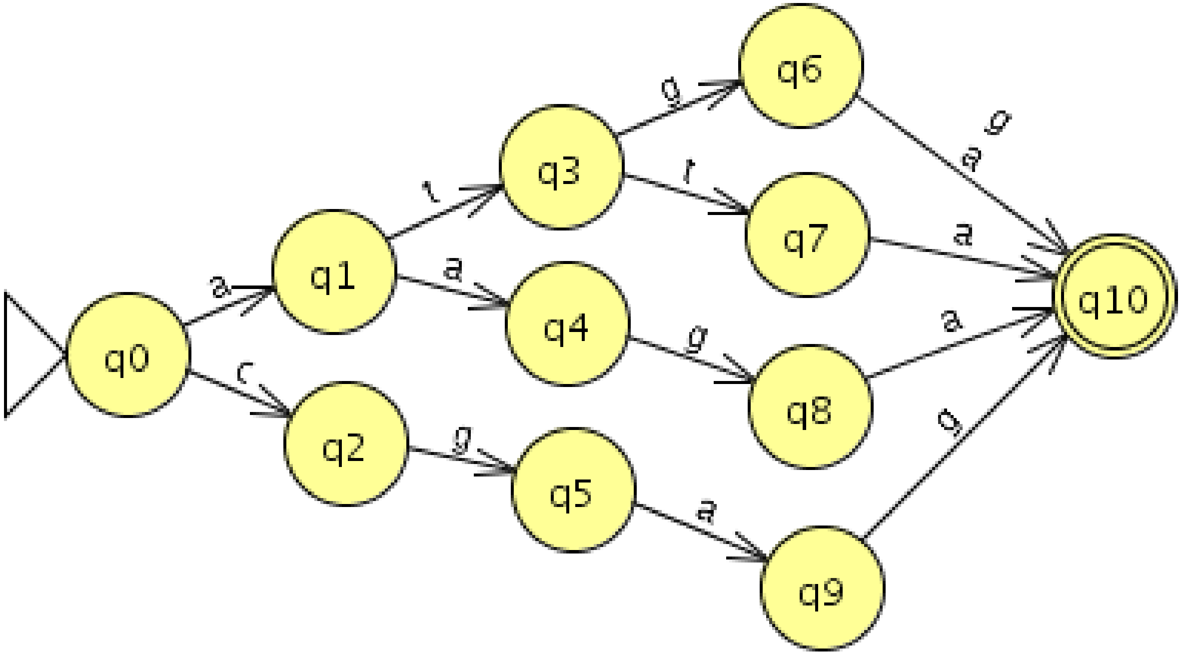
AFDs e AFNs

- Por que o teorema de equivalência é importante?
- Pode-se optar por um ou outro dependendo do objetivo
- AFDs são mais eficientes
- AFNs podem:
  - ser mais fáceis de serem projetados
  - facilitar demonstração de teoremas
  - ser úteis em versões probabilísticas

Exemplo

Este autômato é um AFN !!!

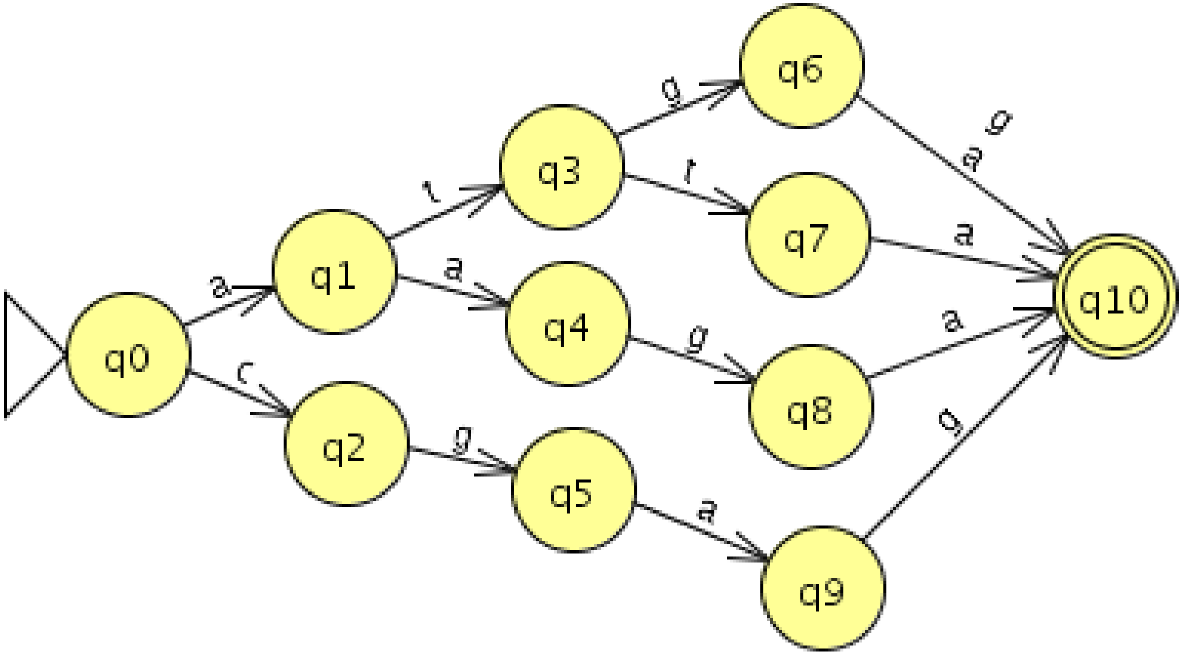
```
atga
atgg
atta
aaga
cgag
```



Exemplo

Este autômato é um AFN !!!

atga
atgg
atta
aaga
cgag



Exercício

Q \ Σ	a	c	g	t
q0				
q1				
q2				
q3				
q4				
q5				
q6				
q7				
q8				
q9				
q10				

AFDs e AFNs

- Por que o teorema de equivalência é importante?
- Pode-se optar por um ou outro dependendo do objetivo
- AFDs são mais eficientes
- AFNs podem:
  - ser mais fáceis de serem projetados
  - facilitar demonstração de teoremas
  - ser úteis em versões probabilísticas

Mais detalhes?
(vídeos de minha disciplina de Introdução a Teoria da Computação - ACH2043)

- AFDs:
  - http://eaulas.usp.br/portal/video.action?idItem=16299
  - http://eaulas.usp.br/portal/video.action?idItem=16300
  - http://eaulas.usp.br/portal/video.action?idItem=16301
- AFNs:
  - https://drive.google.com/file/d/1AAS5FEM9mr_WEnmMPn5egtjngethXHfY/view
- Equivalência entre AFDs e AFNs:
  - http://eaulas.usp.br/portal/video.action?idItem=16901
  - http://eaulas.usp.br/portal/video.action?idItem=16902

Gramáticas regulares e autômatos finitos

- Toda gramática **linear à esquerda** é equivalente a uma gramática **linear à direita** e vice-versa (prova em [RAMOS, 2009]).
- Toda gramática **linear à direita** é equivalente a um **autômato finito**, que é o dispositivo reconhecedor de linguagens regulares (prova mais à frente).
- Enquanto uma gramática regular gera cadeias de uma linguagem L, um autômato finito (para L) é capaz de analisar uma dada cadeia de entrada w e aceitá-la se w $\in$ L ou rejeitá-la caso contrário (classificação).

atga
atgg
atta
aaga
cgag

Exemplo

gramática regular (linear à direita)

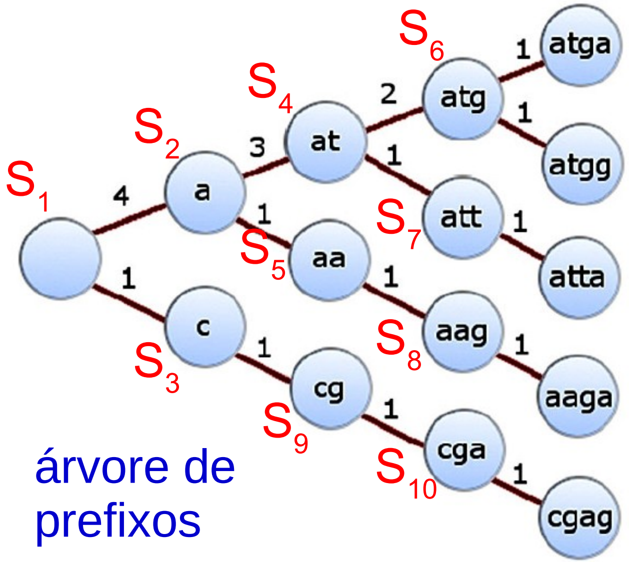
$G = \{V, \Sigma, S, P\}$

$V = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$

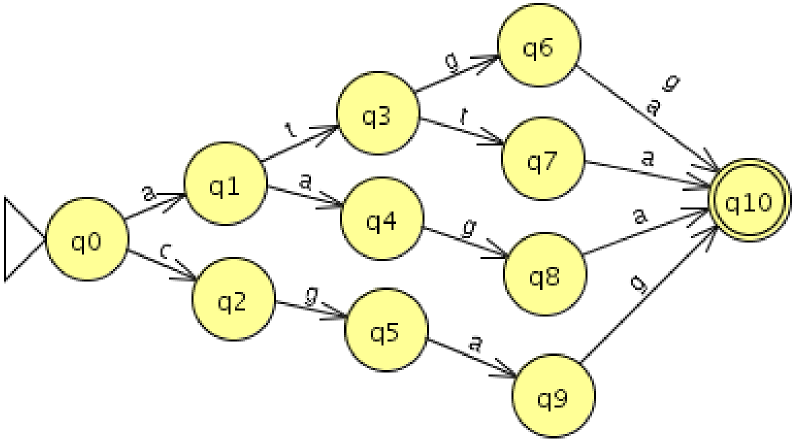
$\Sigma = \{a, c, g, t\}$

$S = S_1$

$P = \{$
$S_1 \rightarrow aS_2 \mid cS_3$
$S_2 \rightarrow tS_4 \mid aS_5$
$S_4 \rightarrow gS_6 \mid tS_7$
$S_6 \rightarrow a \mid g$
$S_7 \rightarrow a$
$S_5 \rightarrow gS_8$
$S_8 \rightarrow a$
$S_3 \rightarrow gS_9$
$S_9 \rightarrow aS_{10}$
$S_{10} \rightarrow g$
$\}$



árvore de prefixos



AFN

Gramáticas regulares e autômatos finitos

- Toda gramática **linear à esquerda** é equivalente a uma gramática **linear à direita** e vice-versa (prova em [RAMOS, 2009]).
- Toda gramática **linear à direita** é equivalente a um **autômato finito**, que é o dispositivo reconhecedor de linguagens regulares (prova mais à frente).
- =>
- <=

Gramáticas lineares à direita => AFN

$G = (V, \Sigma, S, P)$, $M = (Q, \Sigma, q_0, \delta, F)$

$Q = V \cup \{Z\}$, Z não pertence a V

$q_0 = S$

$F = \{Z\}$

$\delta = \ldots$ (vou construir) $\delta \leftarrow \emptyset$

para cada produção em P

se $X \rightarrow aY$, então $\delta \leftarrow \delta \cup \{ (X,a) = Y \}$

se $X \rightarrow Y$, então $\delta \leftarrow \delta \cup \{ (X,\varepsilon) = Y \}$

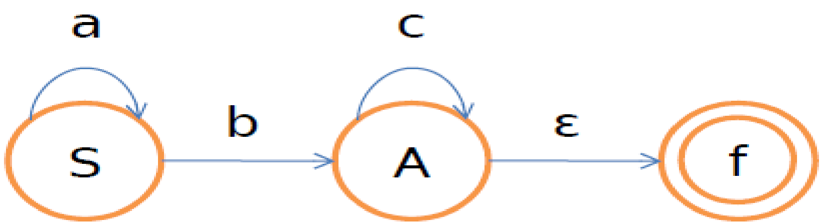
se $X \rightarrow a$, então $\delta \leftarrow \delta \cup \{ (X,a) = Z \}$

se $X \rightarrow \varepsilon$, então $\delta \leftarrow \delta \cup \{ (X,\varepsilon) = Z \}$

Exemplo

$S \rightarrow aS \mid bA$

$A \rightarrow cA \mid \varepsilon$



AFN => Gramáticas lineares à direita

$M = (Q, \Sigma, q_0, \delta, F)$, $G = (V, \Sigma, S, P)$

$V = Q$

$S = q_0$

$P = \ldots$ (vou construir) $P \leftarrow \emptyset$

para cada transição de δ

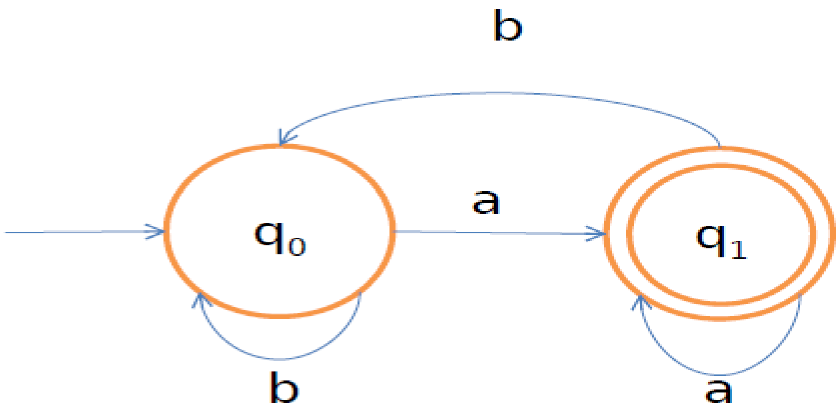
Se $\delta(X,a) = Y$ então $P \leftarrow \{X \rightarrow aY\}$

Se $\delta(X,\varepsilon) = Y$ então $P \leftarrow \{X \rightarrow Y\}$

para cada estado X de F

$P \leftarrow \{X \rightarrow \varepsilon\}$

Exemplo



$$q_0 \rightarrow aq_1 \mid bq_0$$

$$q_1 \rightarrow aq_1 \mid bq_0 \mid \varepsilon$$

Fim do vídeo 4

Autômatos Finitos Não-determinísticos

Professora:
Ariane Machado Lima

Vídeo 5

Comentários Finais

Professora:
Ariane Machado Lima

Gramáticas regulares e autômatos finitos

- Uma linguagem é regular se ela é gerada por uma gramática regular
- Uma gramática é regular se ela for linear à esquerda ou linear à direita
- Toda gramática **linear à esquerda** é equivalente a uma gramática **linear à direita** e vice-versa (prova em [RAMOS, 2009]).
- Toda gramática **linear à direita** é equivalente a um **autômato finito**, que é o dispositivo reconhecedor de linguagens regulares (prova mais à frente).

Linguagens Regulares

DEFINIÇÃO 1.16

Uma linguagem é chamada de uma ***linguagem regular*** se algum autômato finito a reconhece.

TEOREMA 1.39

Todo autômato finito não-determinístico tem um autômato finito determinístico equivalente.

COROLÁRIO 1.40

Uma linguagem é regular se e somente se algum autômato finito não-determinístico a reconhece.

Linguagens Regulares

DEFINIÇÃO 1.16

Uma linguagem é chamada de uma ***linguagem regular*** se algum autômato finito a reconhece.

TEOREMA 1.39

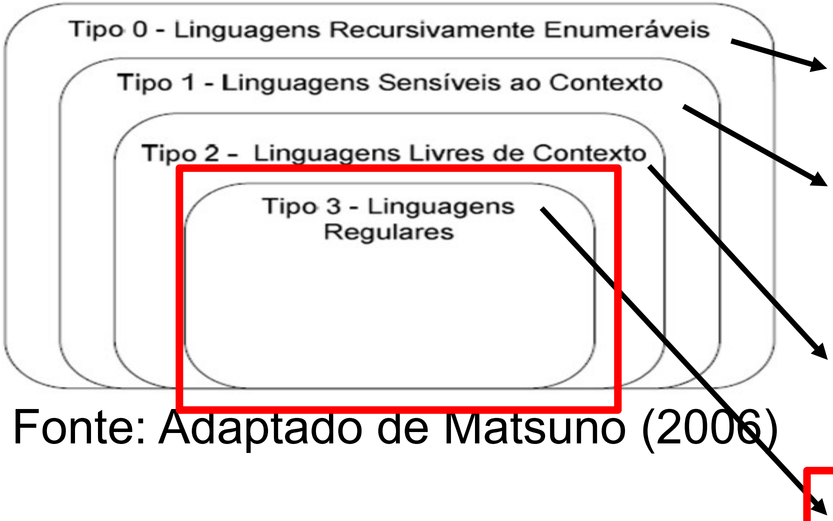
Todo autômato finito não-determinístico tem um autômato finito determinístico equivalente.

COROLÁRIO 1.40

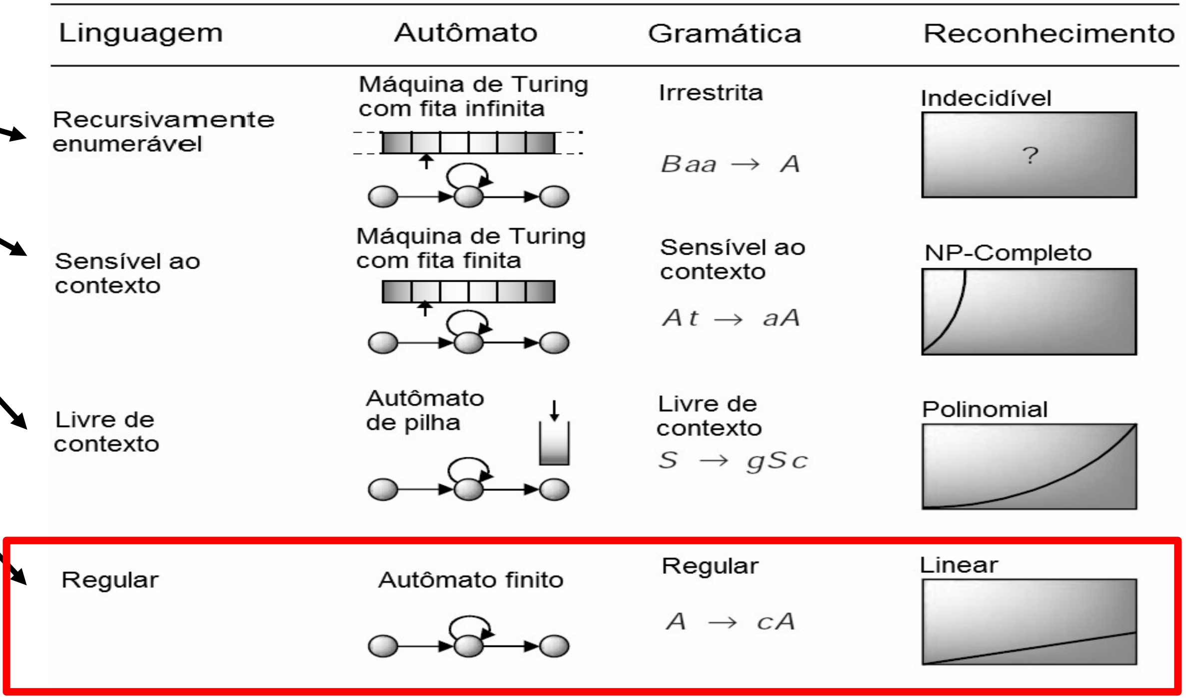
Uma linguagem é regular se e somente se algum autômato finito não-determinístico a reconhece.

E representada por uma expressão regular

Hierarquia de Chomsky



Fonte: Adaptado de Matsuno (2006)



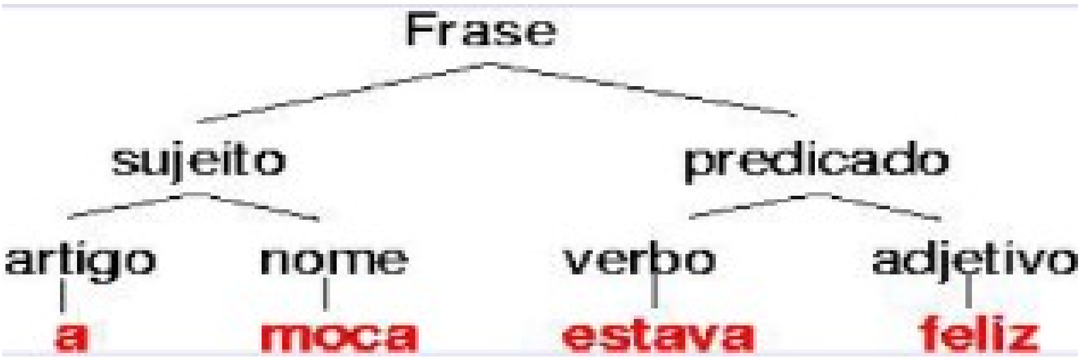
Fonte: Adaptado de Searls (2002)

Linguagens regulares

Mas não se engane! Muitas linguagens são mais que regulares...

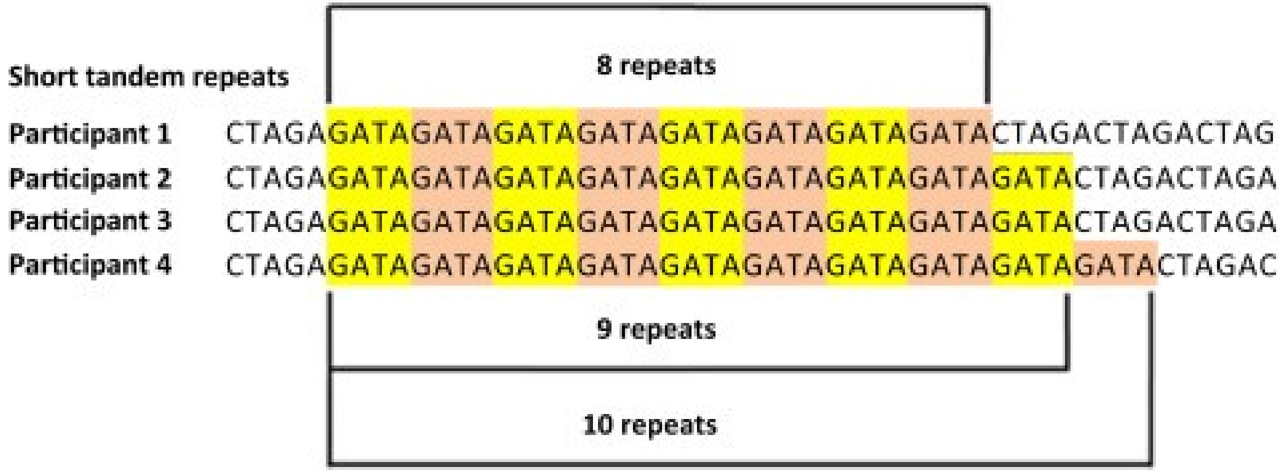
Linguagens regulares

Mas não se engane! Muitas linguagens são mais que regulares...



Linguagens regulares

Mas não se engane! Muitas linguagens são mais que regulares…



https://www.sciencedirect.com/topics/medicine-and-dentistry/short-tandem-repeat

Linguagens regulares

Mas muitas são !

IEICE TRANS. INF. & SYST., VOL.E94–D, NO.5 MAY 2011

PAPER *Special Section on Formal Approach*

Modeling, Verification and Testing of Web Applications Using Model Checker

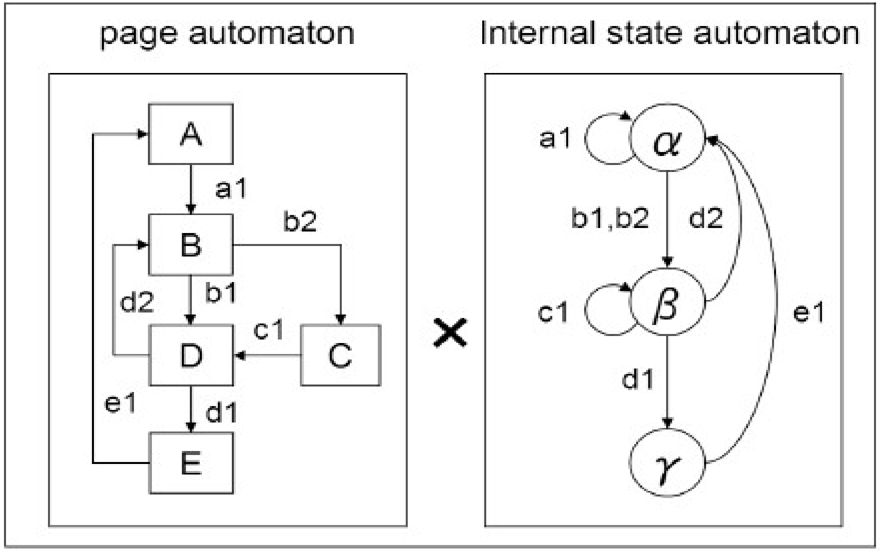


Fig. 1 Model of Web application.

Referências

- RAMOS, M. V. M.; NETO, J. J.; VEJA, I. S. **Linguagens Formais: Teoria, Modelagem e Implementação**. Ed. Bookman, 2009.
- SIPSER, M. **Introdução à Teoria da Computação**. Ed. Thomson, 2007
- DUDA, R.; HART, P.; STORK, D. **Pattern Classification and Scene Analysis**. Ed. John Wiley, 2001. Cap 8.6 e 8.7
- DURBIN, R.; EDDY, S. R.; KROGH, A. **Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.** Cambridge University Press, 2002. Cap 9

Fim do vídeo 5

Comentários Finais

Professora:
Ariane Machado Lima