Universidade de Santa Cruz do Sul – UNISC
Departamento de informática

COMPILADORES

Revisão – Linguagens formais – Parte 01

Geovane Griesang
geovanegriesang@unisc.br

Linguagens formais

Legenda:

$\sum$ = sigma (somatório)
δ = delta
ε = épsilon
λ = lambda
α = alfa
β = beta
γ = gamma
ξ = xi

Linguagens formais

Uma linguagem é um meio de comunicação, formada por um conjunto de palavras e de regras gramaticais que permitem combinar as palavras em sentenças sintaticamente corretas.

Uma linguagem é dita formal quando pode ser representada através de um sistema com sustentação matemática.

A Linguística Formal compreende a representação da sintaxe (estrutura) e da semântica (significado) das sentenças de uma linguagem.

Linguagens formais

Alfabeto $\sum$, ou vocabulário, é um conjunto finito (não vazio) de símbolos.

Exemplo: dígitos, letras, letras gregas, ...

Uma sentença, ou palavra, ou cadeia em um alfabeto $\sum$ é uma sequencia finita de símbolos deste alfabeto.

Exemplo alfabetos:
$\sum_1$={0, 1}
$\sum_2$={a, b, c, ..., z}

Exemplo cadeias:
01001
abracadabra

Linguagens formais

O tamanho, ou comprimento, de uma sentença é dado pelo número de símbolos que compõem a sentença.

Exemplo:

alfabeto V = {a, b, c, ..., z}
sentença w = abracadabra
tamanho |w| = 11

Sentença vazia é a sentença que não contém símbolos, possuindo tamanho 0. É representada por ε.

Linguagens formais

Seja V um alfabeto, representamos por:

V^* = conjunto de todas as sentenças compostas de símbolos de V incluindo a sentença vazia.

$V^+ = V^* - (\varepsilon)$.

Exemplos:

V = { 0,1}

V^* = { ε, 0, 1, 00, 01, 10, 11, ...}

V^+ = { 0, 1, 00, 01, 10, 11, ...}

Linguagens formais

Então, Linguagem é qualquer conjunto de sentenças sobre um alfabeto, ou seja, uma linguagem L sobre um alfabeto V é um subconjunto de V*.

Portanto, conforme o número de sentenças que possuem, as linguagens se classificam em:

- ✓ Finitas
- ✓ Vazias
- ✓ Infinitas

Linguagens formais

Como representar uma linguagem?

1. listando as palavras (só para linguagem finita).

2. através de uma descrição algébrica. Ex: (a^n b^n c^n | n >= 1).

3. definindo um algoritmo que determina se uma sentença pertence ou não à linguagem: Reconhecimento.
   Exemplo: autômato finito.

4. definindo um mecanismo que gera sistematicamente as sentenças da linguagem em alguma ordem: Geração.
   Exemplo: gramática.

Linguagens formais

Uma gramática serve para definir qual o subconjunto de sentenças que faz parte de uma determinada linguagem.

Ela é um dispositivo formal para especificar uma linguagem potencialmente infinita de uma forma finita.

Linguagens formais

Uma gramática G é definida por uma quádrupla:

$G = (N, T, P, S)$ onde,

- N - conjunto finito de não-terminais
- T - conjunto finito de terminais
- P - conjunto finito de regras de produção
- S - símbolo inicial da gramática

Observações: $N \cap T = \varnothing$

$V = N \cup T$

$S \in N$

$P = \{ \alpha \rightarrow \beta \mid \alpha \in V^{+} \text{ e } \beta \in V^{*} \}$

Linguagens formais

N - conjunto finito de não-terminais

T - conjunto finito de terminais

P - conjunto finito de regras de produção

S - símbolo inicial da gramática

Exemplo:

$G = (N, T, P, I)$

$N = \{I, D\}$

$T = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$

$P = \{ I \rightarrow D , D \rightarrow 0 , D \rightarrow 1 , D \rightarrow 2 , D \rightarrow 3 , D \rightarrow 4 ,$
$D \rightarrow 5 , D \rightarrow 6 , D \rightarrow 7 , D \rightarrow 8 , D \rightarrow 9 \}$

Linguagens formais

Convenções: Para facilitar a compreensão das expressões, são adotadas sempre que possível as seguintes convenções:

N = {A, B, C, ..., T}: Letras maiúsculas do início do alfabeto para símbolos não-terminais

T = {a, b, c, ..., t}: Letras minúsculas do início do alfabeto para símbolos terminais

T^* = {u, v, ..., z}: Letras minúsculas do fim do alfabeto para sequências de terminais

Linguagens formais

Convenções

$(N \cup T)^*$ = {α, β, γ, ...}: Letras gregas para sequências de variáveis e terminais

$N \cup T$ = {U, V, ... Z}: Letras maiúsculas do fim do alfabeto para um símbolo terminal ou não-terminal

Linguagens formais

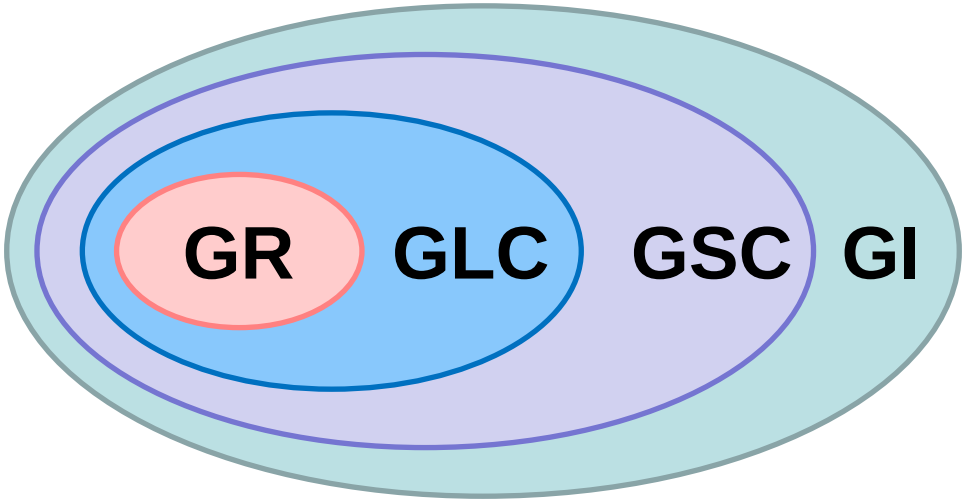
Tipos de gramática

Tipo 0: Gramáticas Irrestritas (GI).

Tipo 1: Gramáticas Sensíveis ao Contexto (GSC).

Tipo 2: Gramáticas Livres de Contexto (GLC).

Tipo 3: Gramáticas Regulares (GR).



Linguagens formais

N – conj. finito de não-terminais
T – conj. finito de terminais
P – conj. finito de regras de produção
S – símbolo inicial da gramática
$N \cap T = \varnothing$ $V = N \cup T$
$P = \{ \alpha \rightarrow \beta \mid \alpha \in V^+ \text{ e } \beta \in V^* \}$

Tipo 0: Gramáticas Irrestritas (GI)

Nenhuma limitação é imposta.

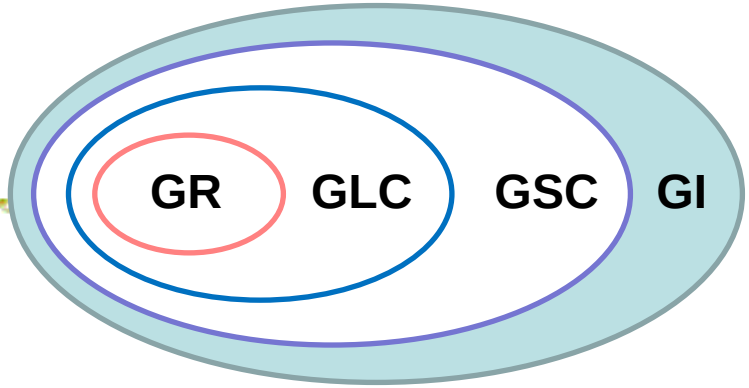
São definidas pelas seguintes regras de produção:

$$P = \{ \alpha \rightarrow \beta \mid \alpha \in V^+ , \beta \in V^* \}$$

Do lado esquerdo da produção pode haver uma sequência de quaisquer símbolos, desde que haja um não-terminal.

Do lado direito da produção pode haver qualquer sequência de símbolos, inclusive a sentença vazia.

Linguagens formais



Tipo 0: Gramáticas Irrestritas (GI)

Ex.: L = $\{a^n b^{2n} a^n / n \geq 1\}$

S	→	aAbba
aAb	→	aabbbA | ab
bAb	→	bbA
bAa	→	Bbaa
bB	→	Bb
aB	→	aA

A cadeia aaabbbbbbaaa pode ser derivada dessa gramática:

S => aAbba	=> aabbbAba
=> aabbbbAa	=> aabbbBbaa
=> aabbBbbaa	=> aabBbbbaa
=> aaBbbbbaa	=> aaAbbbbaa
=> aaabbbAbbbaa	=> aaabbbbAbbaa
=> aaabbbbbAbaa	=> aaabbbbbbAaa
=> aaabbbbbBbaaa	=> aaabbbbBbbaaa
=> aaabbbBbbbaaa	=> aaabbBbbbbaaa
=> aaabBbbbbbaaa	=> aaaBbbbbbbaaa
=> aaaAbbbbbbaaa	=> aaabbbbbbaaa

Linguagens formais

N – conj. finito de não-terminais
T – conj. finito de terminais
P – conj. finito de regras de produção
S – símbolo inicial da gramática
$N \cap T = \varnothing$ $V = N \cup T$
$P = \{ \alpha \rightarrow \beta \mid \alpha \in V^+ \text{ e } \beta \in V^* \}$

Tipo 1 - Gramáticas Sensíveis ao Contexto (GSC)

Para toda produção: $\alpha \rightarrow \beta \in P$

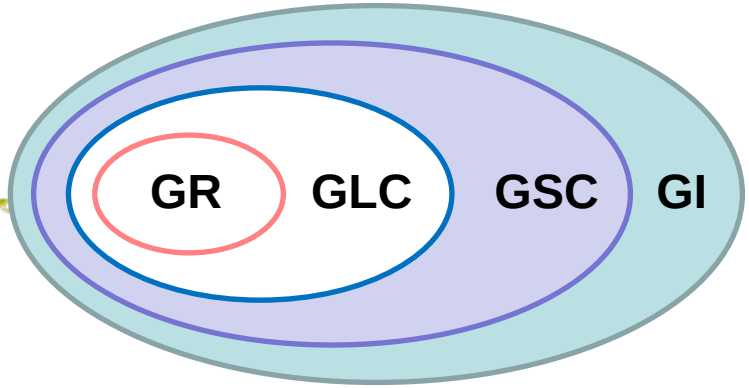
$|\alpha| \leq |\beta|$

Comprimento da sentença do lado esquerdo deve ser menor ou igual ao comprimento da sentença do lado direito.

Do lado direito não é aceito a sentença vazia.

Exemplo: $\alpha_1 A \alpha_2 \rightarrow \alpha_1 B \alpha_2$

Linguagens formais



GI x GSC

Ex.: L = $\{a^n b^{2n} a^n / n \geq 1\}$

Essa GI é GSC?

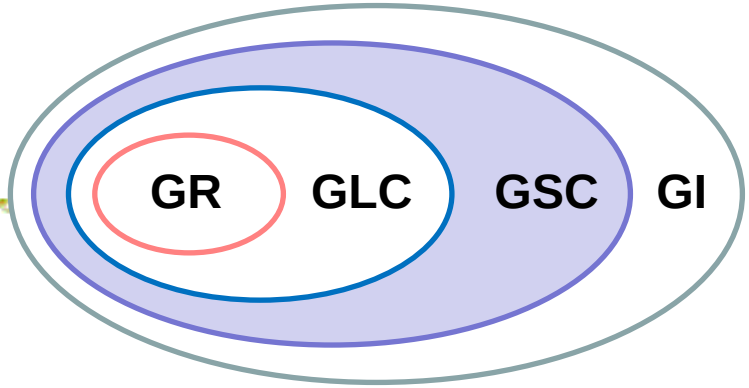
S → aAbba
aAb → aabbbA | ab
bAb → bbA
bAa → Bbaa
bB → Bb
aB → aA

Não!

aAb → ab

O lado direito da produção é menor que o lado esquerdo.

Linguagens formais



Tipo 1 - GSC

Ex.: L = $\{a^n b^{2n} a^n / n \geq 1\}$

S	→	aAbba | abba
aAb	→	aabbbB
Bb	→	bB
Ba	→	Caa | aa
bCa	→	Cba
bC	→	Cb
aCb	→	aAb

A cadeia aaabbbbbbaaa pode ser derivada:

S	=> aAbba	=> aabbbBba
	=> aabbbbBa	=> aabbbbCaa
	=> aabbbCbaa	=> aabbCbbaa
	=> aabCbbbaa	=> aaCbbbbaa
	=> aaAbbbbaa	=> aaabbbBbbbaa
	=> aaabbbbBbbaa	=> aaabbbbbBbaa
	=> aaabbbbbbBaa	=> aaabbbbbbaaa

Linguagens formais

N – conj. finito de não-terminais
T – conj. finito de terminais
P – conj. finito de regras de produção
S – símbolo inicial da gramática
$N \cap T = \varnothing$ $V = N \cup T$
$P = \{ \alpha \rightarrow \beta \mid \alpha \in V^+ \text{ e } \beta \in V^* \}$

Tipo 2 - Gramáticas Livres de Contexto (GLC)

Quando as regras de produção são todas na seguinte forma:

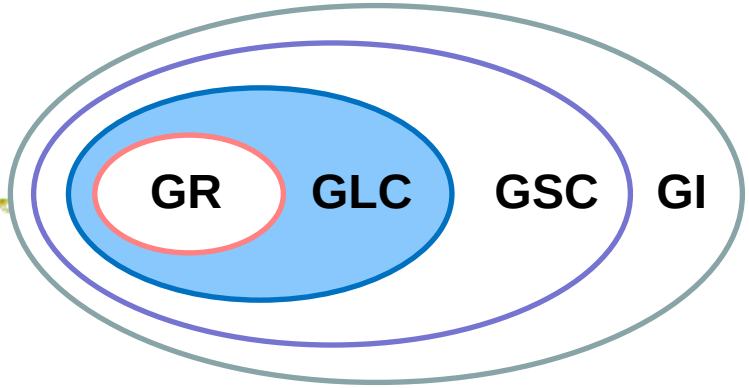
$$P = \{ \alpha \rightarrow \beta \mid \alpha \in N \text{ e } \beta \neq \varepsilon \}$$

Do lado esquerdo da produção deve, sempre, ocorrer um e apenas um não-terminal. A sentença vazia também não é aceita do lado direito da produção.

Exemplo: $X \rightarrow abcX$

{não interessa o contexto em que X se encontra}

Linguagens formais



Tipo 2 - GLC

Em Ciência da computação é importante para descrição da estrutura de Linguagens de programação.

Exercício 01: $L = \{a^n b^{2n} / n \geq 1\}$.

Gerar uma gramática livre de contexto que possa gerar a seguinte cadeia aaabbbbbb.

Utilizar o sistema *GlcTransformer*.

Linguagens formais

N – conj. finito de não-terminais
T – conj. finito de terminais
P – conj. finito de regras de produção
S – símbolo inicial da gramática
$N \cap T = \varnothing$ $V = N \cup T$
$P = \{ \alpha \rightarrow \beta \mid \alpha \in V^+ \text{ e } \beta \in V^* \}$

Tipo 3 - Gramáticas Regulares (GR)

Toda produção é da forma:

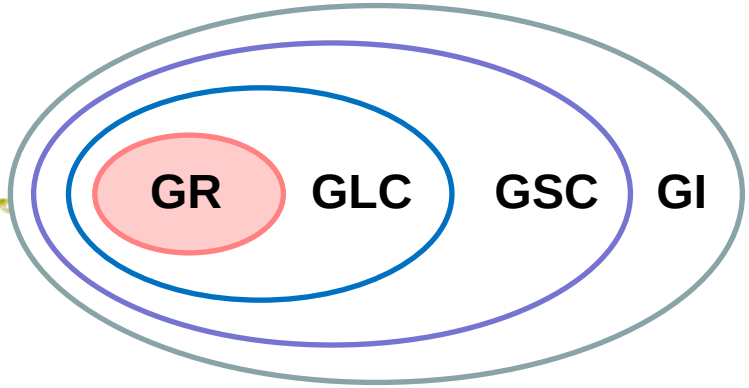
$A \rightarrow aB$ ou $A \rightarrow a$

Ou seja:

$P = \{ A \rightarrow aX \mid A \in N, a \in T, X \in N \cup \{\varepsilon\} \}$

Do lado esquerdo deve, sempre, ocorrer um e apenas um não-terminal e do lado direito podem ocorrer ou somente um terminal, ou um terminal seguido de um não-terminal.

Linguagens formais



Tipo 1 - GR

Ex.: L = $\{a^n b \,/\, n \geq 1\}$

Exercício 02: Transforme a GLC abaixo em uma GR.

S	→ AB
A	→ aA | a
B	→ b

Linguagens formais

Linguagens geradas por gramáticas:

Portanto, conforme o tipo da gramática que dá origem a uma linguagem, estas se classificam em:

LSC - Linguagem Sensível ao Contexto
LLC - Linguagem Livre de Contexto
LR - Linguagem Regular

Linguagens formais

A sentença vazia

Vamos estender as definições das gramáticas Tipo 1, 2 e 3 para permitir produções do tipo $S \rightarrow \varepsilon$, sendo S o símbolo inicial.

Entretanto, isto só será possível se S não aparecer do lado direito de qualquer produção.

Desde modo, a regra $S \rightarrow \varepsilon$ somente será utilizada para dar origem à sentença vazia.

Linguagens formais

A sentença vazia

A gramática que possuir o símbolo inicial aparecendo do lado direito de regras de produção, deverá ser transformada em outra equivalente que obedeça a esta restrição.

Observação: gramáticas equivalentes devem gerar a mesma linguagem.

Linguagens formais

A sentença vazia

Quando for necessário transformar a gramática, deverá ser incluído um novo símbolo não-terminal (S') que passará a ser o novo símbolo inicial.

Em seguida, deverão ser incluídas em P todas as regras que tinham o símbolo S do lado esquerdo, substituindo este por S', mais a regra $S' \rightarrow \varepsilon$.

Linguagens formais

A sentença vazia. Exemplo: Modificar a gramática abaixo de modo a incluir a sentença vazia.

$$G = \{ \{ S, A \}, \{ a, b\}, P, S \}$$

$$P = \{ \quad S \rightarrow aS \mid aA$$

$$A \rightarrow bA \mid b \}$$

Resposta:

$$G = \{ \{ S', S, A \}, \{ a, b\}, P', S' \}$$

$$P' = \{ \quad S' \rightarrow S \mid \varepsilon$$

$$S \rightarrow aS \mid aA$$

$$A \rightarrow bA \mid b \}$$

Linguagens formais

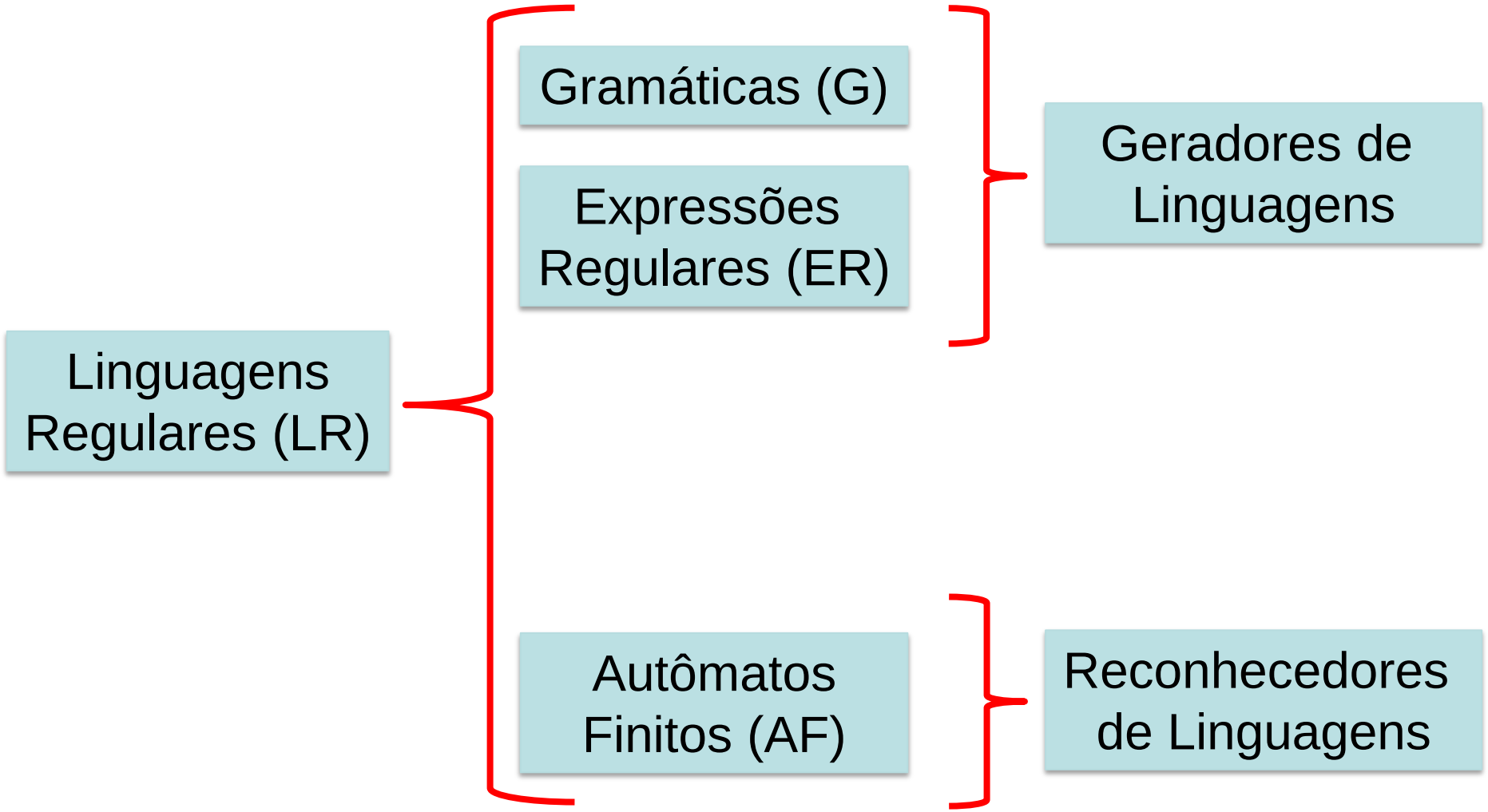
Expressões regulares (ER)

Toda linguagem regular pode ser descrita por uma expressão simples, denominada Expressão Regular (ER).

Trata-se de um formalismo gerador, pois expressa como construir (gerar) as palavras da linguagem.

Uma ER é definida recursivamente a partir de conjuntos (linguagens) básicos e operação de concatenação e união.

Linguagens formais



Linguagens formais

Regras de formação de expressões regulares

Dado um alfabeto:

1) os símbolos do alfabeto são expressões regulares;
2) se R1 e R2 são ER, então (R1 $\cup$ R2) é uma ER;
3) se R1 e R2 são ER, então (R1 R2) é uma ER;
4) se R1 é uma ER, então $(R1)^*$ é uma ER;

(R1 R2) representa concatenação de linguagens;

$(R1)^*$ representa a linguagem formada pela concatenação de zero ou mais palavras de R1.

(R1|R2) representa união de linguagens

Linguagens formais

Observação: p+ = pp*

Expressão Regular	Linguagem Gerada
aa	contém somente a palavra aa
ba*	todas as palavras que iniciam por b, seguido de zero ou mais a's
(a) *	{ ε, a, aa, aaa, ... }
(a | b)*	{ ε, a, b, aa, ab, bb, ba, aaa, ... }
(a (a | b))*	{ ε, aa, ab, aaaa, abaa, aaab, ... }
(a (a | b)*)	{ a, aa, ab, aaa, aba, aab, ... }
Identificadores: l (l | d)*	{ l, ll, ld, lll, ... }
Inteiro com sinal (+ | -) d (d)*	{ +d, -d, +dd, -dd, ... }

Linguagens formais

Autômatos Finitos (AT)

É um modelo matemático (máquina abstrata) de um sistema, com entradas e saídas discretas.

O sistema pode estar em qualquer uma de um número finito de configurações internas (ou *estados*).

O estado de um sistema resume as informações anteriores até o momento atual do reconhecimento necessárias para determinar o seu comportamento face ao resto da sequência que está analisando.

Linguagens formais

Autômatos Finitos (AT)

Exemplo: mecanismo de controle de um elevador.

Ele não lembra todas as requisições anteriores mas somente o andar atual, a direção de movimento (para cima ou para baixo), e a relação de requisições pendentes.

Linguagens formais

Autômato Finito Determinístico (AFD)

É definido formalmente por uma 5-tupla (K, Σ, δ, q_0, F), onde:

K	conjunto finito, não vazio, de estados
Σ	alfabeto finito de entrada
q_0	estado inicial e $q_0 \in K$
F	conjunto de estados finais e $F \subseteq K$
δ	função de transição de estados, mapeando $K \times \Sigma$ em K.

$\delta (q, a)$ é um estado p/ cada estado q e símbolo de entrada a.

Linguagens formais

Autômato Finito Determinístico (AFD)

Um autômato finito é representado através de um controle finito, que tem acesso a uma fita onde está a sequência a ser analisada.

O autômato percorre esta fita da esquerda para a direita, lendo um símbolo de cada vez.

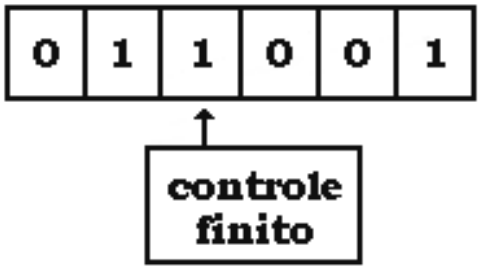
Estando o controle em um estado, apontando para um determinado símbolo na entrada, ocorre uma transição de estado, que faz com que o controle passe para outro estado e avance para o próximo símbolo na entrada.

Linguagens formais

Autômato Finito Determinístico (AFD)

Se isto se repetir com sucesso até o final da fita de entrada e, no final, o autômato estiver em um estado final, a sequência foi reconhecida.

Se, ao contrário, ocorrer alguma falha (transição impossível) o reconhecimento para e a sequência não é reconhecida.



Linguagens formais

Diagramas de transição

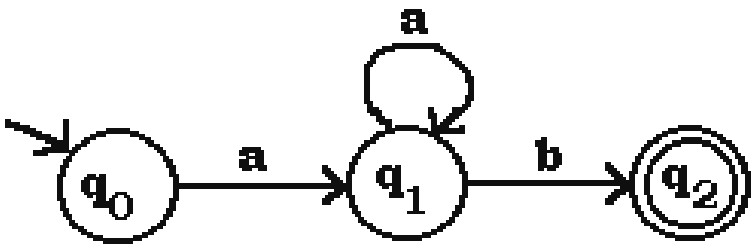
Uma outra maneira de representar um autômato finito são os diagramas de transição. Trata-se de um grafo direcionado e rotulado.

Os vértices representam os estados, sendo representados por círculos. Os estados finais são representados por 2 (dois) círculos concêntricos e o estado inicial é indicado por uma seta. As arestas representam as transições entre 2 estados, sendo o rótulo o símbolo reconhecido.

Linguagens formais

Diagramas de transição

Exemplo



M = (estados, transições, δ, E. ini, E. fin.)

$\delta(q_0, a) = q_1$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_2$

M = (K, Σ, δ, q_0, F)

M = ((q_0, q_1, q_2), { a, b }, δ, q_0, {q_2})

M = ({ q0,q1,q2 } , { a, b } , S , q0 , { q2 })

T (M) = { a^n b | n >= 1 }

Linguagens formais

Tabela de Transição de Estados

Uma terceira maneira de representar um autômato finito é através de uma tabela, indicando na vertical os estados e na horizontal os símbolos do alfabeto. O estado inicial é indicado com uma seta e os estados finais com asteriscos. O conteúdo da tabela indica as transições de estado possíveis.

		a	b
→	q_0	q_1	-
	q_1	q_1	q_2
*	q_2	-	-

Linguagens formais

Extensão da função δ para o domínio K x Σ^*

$\xi\ (q, \varepsilon) = q$

$\xi\ (q, xa) = \delta\ (\xi\ (q, x), a)$, para $x \in \Sigma^*$ e $a \in \Sigma$

$\xi\ (q, x) = p$ significa que M, iniciando no estado q, com a sequência x na fita de entrada, entrará no estado p, após o cabeçote mover-se para a direita tantas células quanto for o comprimento de x.

Uma sentença x é aceita por M se $\delta\ (q_0, x) = p$ para algum p em F.

Linguagens formais

Linguagem aceita por M

O conjunto de todos os x aceitos por M é:

$$T(M) = \{ x \mid \delta (q_0, x) = p \wedge p \in F \}$$

Qualquer conjunto de sentenças aceito por um autômato finito é dito ser REGULAR.

Linguagens formais

Autômato Finito Não-Determinístico (AFND)

Um AFND é um sistema (K, $\sum$, δ, q_0, F), onde:

K conjunto finito, não vazio, de estados

$\sum$ alfabeto finito de entrada

q_0 estado inicial e $q_0 \in K$

F conjunto de estados finais e $F \in K$

δ função de transição de estados, mapeando K x $\sum$ em 2^K.

$\delta(q, a)$: subconjunto de K para cada estado *q* e símbolo de entrada *a*. 2^K é o conjunto de todos os subconjuntos de K.

Linguagens formais

Diferença importante entre AFD e AFND

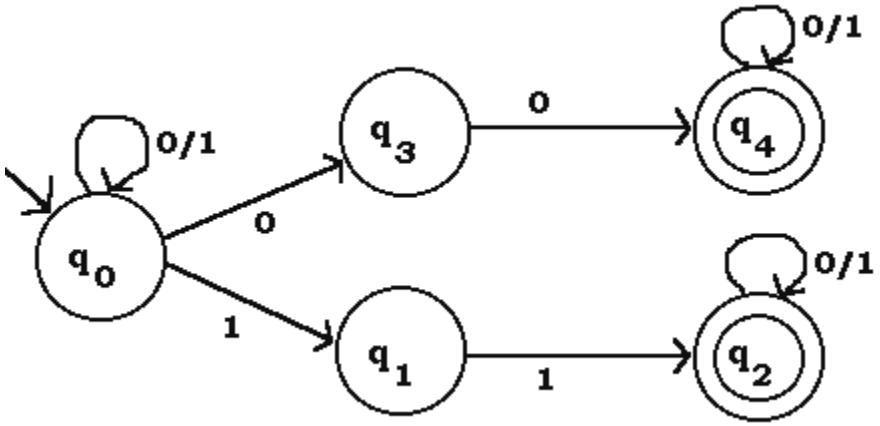
$\delta(q, a)$ é um conjunto de estados (possivelmente vazio) ao invés de um único estado.

$\delta(q, a) = \{ p_1, p_2, ..., p_k \}$, significa que, M estando no estado q, olhando a na fita de entrada, move uma célula para a direita e escolhe qualquer um de $p_1, p_2, ..., p_k$ como próximo estado.

Exemplo: AFND que aceita o conjunto de todas as sentenças que contém dois 0's ou dois 1's consecutivos.

Linguagens formais

AFND



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

$\delta(q_0, 0) = \{ q_0, q_3 \}$ $\quad$ $\delta(q_0, 1) = \{ q_0, q_1 \}$

$\delta(q_1, 0) = \varnothing$ $\quad$ $\delta(q_1, 1) = \{ q_2 \}$

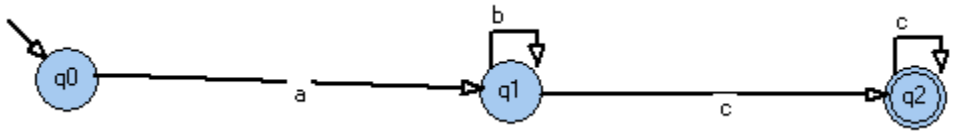
$\delta(q_2, 0) = \{ q_2 \}$ $\quad$ $\delta(q_2, 1) = \{ q_2 \}$

$\delta(q_3, 0) = \{ q_4 \}$ $\quad$ $\delta(q_3, 1) = \varnothing$

$\delta(q_4, 0) = \{ q_4 \}$ $\quad$ $\delta(q_4, 1) = \{ q_4 \}$

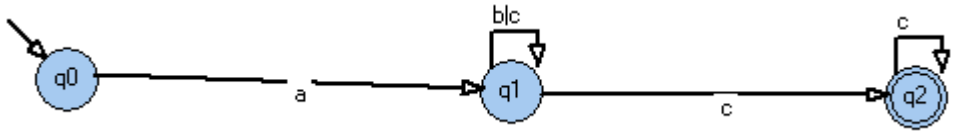
Linguagens formais

AFD x AFND



	a	c	b
=> q0	q1		
q1		q2	q1
* q2		q2	

Informações
Total Transições = 4
Tipo Autômato = AFD
Total Estados = 3



Informações
Total Transições = 4
Tipo Autômato = AFND
Total Estados = 3

	a	c	b
=> q0	q1		
q1		q2,q1	q1
* q2		q2	

Linguagens formais

AFD x AFND

Se L é um conjunto aceito por um AFND, então existe um AFD que aceita L.

Ou seja, um AFND pode ser transformado em um AFD equivalente.

Linguagens formais

AFND => AFD

M’ = (K’, Σ‘, δ‘, q_0’, F’) tal que:

Os estados de M’ constituem todos os subconjuntos do conjunto de estados de M: K’ = 2^K.

F’ é o conjunto de todos os estados de K’ contendo um estado de F.

Linguagens formais

AFND => AFD

um elemento de K' é denotado por [q_1 q_2 ... q_i] onde q_1 q_2 ... q_i $\in$ K.

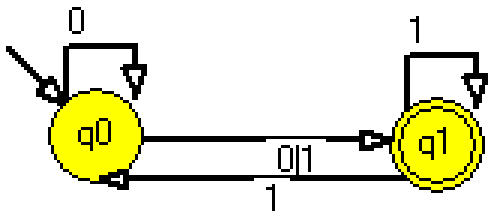
q_0' = [q_0]

definimos δ' ([q_1 q_2 ... q_i], a) = [p_1 p_2 ... p_j]

se e somente se δ ({ q_1, q_2, ... q_i }) = { p_1, p_2, ... p_j } =

= δ (q_1, a) $\cup$ δ (q_2, a) $\cup$ ... $\cup$ δ (qi, a)

Linguagens formais



AFND => AFD: Construir um AFD M' que reconheça a mesma linguagem que M.

Encontrar M'

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

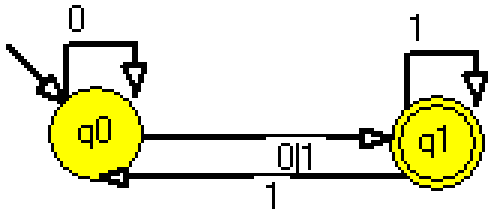
$\delta (q_0, 1) = \{ q_1 \}$

$\delta (q_1, 0) = \varnothing$

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

	0	1
=> q0	q0,q1	q1
* q1		q0,q1

Linguagens formais



AFND => AFD

δ‘ ([q_0, q_1], 0) = [q_0, q_1] desde que:

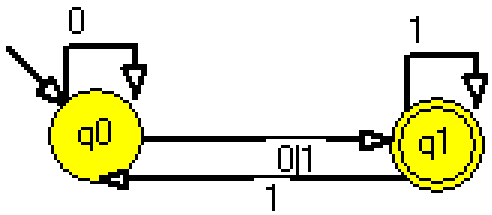
δ ({q_0, q_1 }, 0) = δ (q_0, 0) $\cup$ δ (q_1, 0) = {q_0, q_1} $\cup$ $\varnothing$ = {q_0, q_1}, e

δ‘ ([q_0, q_1], 1) = [q_0, q_1] desde que:

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

δ ($\varnothing$, 0) = δ‘ ($\varnothing$, 0) = $\varnothing$

Linguagens formais



AFND => AFD: Construir um AFD M' que reconheça a mesma linguagem que M.

Encontrar M'

	0	1
=> q0	q0, q1	q1
* q1		q0, q1

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

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

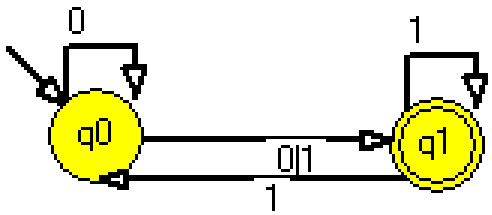
$\delta (q_1, 0) = \varnothing$

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

$\delta' ([q_0], 0) = ?$

$\delta' ([q_0], 1) = ?$

Linguagens formais



AFND => AFD: Construir um AFD M' que reconheça a mesma linguagem que M.

Encontrar M'

	0	1
=> q0	q0, q1	q1
* q1		q0, q1

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

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

$\delta (q_1, 0) = \varnothing$

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

$\delta' ([q_0], 0) = [q_0\ q_1]$

$\delta' ([q_0], 1) = [q1]$

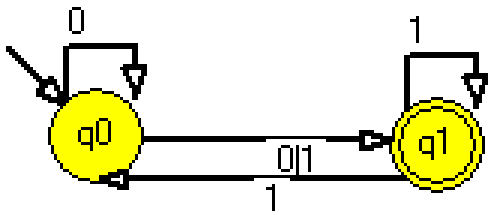
$\delta'' ([q_1], 0) = ?$

$\delta'' ([q_1], 1) = ?$

$\delta''' ([q_0\ q_1], 0) = ?$

$\delta''' ([q_0\ q_1], 1) = ?$

Linguagens formais



AFND => AFD: Construir um AFD M' que reconheça a mesma linguagem que M.

Encontrar M'

	0	1
=> q0	q0, q1	q1
* q1		q0, q1

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

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

$\delta (q_1, 0) = \varnothing$

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

$\delta' ([q_0], 0) = [q_0\ q_1]$

$\delta' ([q_0], 1) = [q1]$

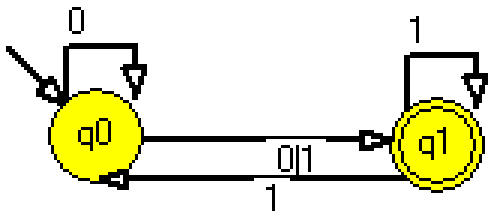
$\delta'' ([q_1], 0) = \varnothing$

$\delta'' ([q_1], 1) = [q_0\ q_1]$

$\delta''' ([q_0\ q_1], 0) = ?$

$\delta''' ([q_0\ q_1], 1) = ?$

Linguagens formais



AFND => AFD: Construir um AFD M' que reconheça a mesma linguagem que M.

Encontrar M'

	0	1
=> q0	q0, q1	q1
* q1		q0, q1

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

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

$\delta (q_1, 0) = \varnothing$

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

$\delta' ([q_0], 0) = [q_0\ q_1]$

$\delta' ([q_0], 1) = [q1]$

$\delta'' ([q_1], 0) = \varnothing$

$\delta'' ([q_1], 1) = [q_0\ q_1]$

$\delta''' ([q_0\ q_1], 0) = [q_0\ q_1]$

$\delta''' ([q_0\ q_1], 1) = [q_0\ q_1]$

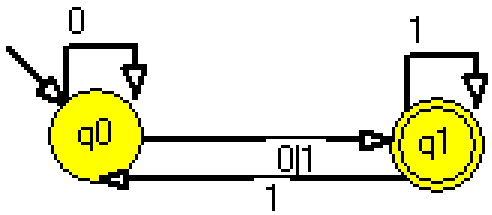
Linguagens formais

AFND => AFD

	δ	0	1
→	q0	[q0, q1]	q1
*	q1	∅	[q0,q1]

	δ‘	0	1
→	[q0]	[q0,q1]	[q1]
*	[q1]	∅	[q0,q1]
*	[q0,q1]	[q0,q1]	[q0,q1]

Linguagens formais



AFND => AFD: Construir um AFD M' que reconheça a mesma linguagem que M.

Encontrar M'

$\delta'([q_0], 0) = [q_0\ q_1]$

$\delta'([q_0], 1) = [q1]$

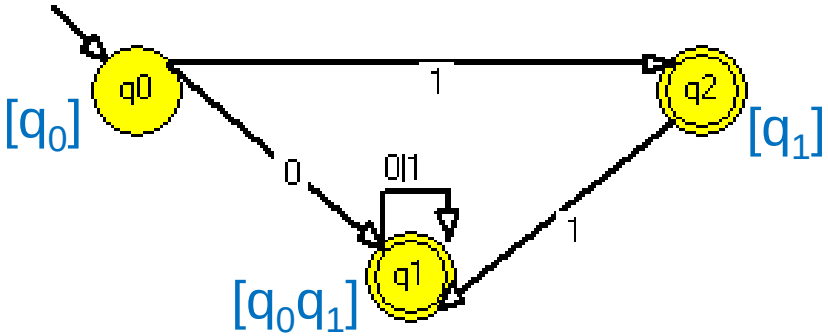
$\delta'([q_1], 0) = \varnothing$

$\delta'([q_1], 1) = [q_0\ q_1]$

$\delta'([q_0\ q_1], 0) = [q_0\ q_1]$

$\delta'([q_0\ q_1], 1) = [q_0\ q_1]$

	0	1
=> [q0]	[q0q1]	[q1]
* [q0q1]	[q0q1]	[q0q1]
* [q1]		[q0q1]



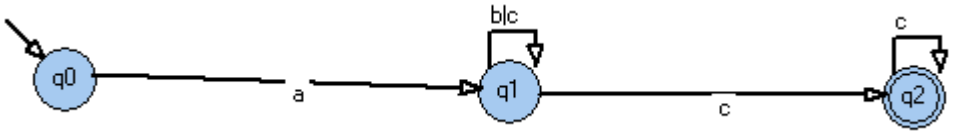
Linguagens formais

Exercício 03: Transformar o AFND abaixo em um AFD.

Depois disto, apresentar o resultado nos seguintes formatos:

Diagrama de transição em forma de grafos.

Diagrama de transição em forma de tabela.



Informações
Total Transições = 4
Tipo Autômato = AFND
Total Estados = 3

	a	c	b
=> q0	q1		
q1		q2,q1	q1
* q2		q2	

Linguagens formais

Exercício 04:

Autômato finito que reconheça sentenças em {0, 1}, as quais não contém dois ou mais 1's consecutivos.

Representar nas seguintes formas:

Diagrama de transição em forma de grafos.
Diagrama de transição em forma de tabela.

Linguagens formais

Exercício 05: utilizando a ferramenta Sagla, verifique se é possível reconhecer as sentenças abaixo (exercício anterior):

Sentença:	Reconhece?	
a) 11	( ) Sim	( ) Não
b) 1100	( ) Sim	( ) Não
c) 10000111	( ) Sim	( ) Não
d) 101010	( ) Sim	( ) Não
e) ε	( ) Sim	( ) Não
f) 001100	( ) Sim	( ) Não
g) 010110	( ) Sim	( ) Não
h) 110011001001	( ) Sim	( ) Não
i) 0	( ) Sim	( ) Não

Linguagens formais

Exercício 06: Construa um AFND que reconheça sentenças sobre o alfabeto {a, b} que iniciem pela letra “b” e possuam o símbolo “a” como penúltimo símbolo. Depois, determine-o.

Construir o autômato no Sagla.

Ex. de sentenças válidas: bab, baa, baaa, baab, bbaa, bbab.

COMPILADORES

Obrigado!!

Geovane Griesang
geovanegriesang@unisc.br