ATITUS
EDUCAÇÃO

Nomes: Wesley Barbaro - 1134832, Wendel Barbaro 1134440

Linguagens Formais e Autômatos
Prof. Me. Fahad Kalil

REVISÃO G1

Hierarquia de Chomsky

Grammar	Languages	Recognizing Automaton	Production rules (constraints)*	Examples[5][6]
Type-3	Regular	Finite-state automaton	$A \to \mathrm{a}$ $A \to \mathrm{a}B$ (right regular) or $A \to \mathrm{a}$ $A \to B\mathrm{a}$ (left regular)	$L = \{a^n \mid n > 0\}$
Type-2	Context-free	Non-deterministic pushdown automaton	$A \to \alpha$	$L = \{a^n b^n \mid n > 0\}$
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A \beta \to \alpha \gamma \beta$	$L = \{a^n b^n c^n \mid n > 0\}$
Type-0	Recursively enumerable	Turing machine	$\gamma \to \alpha$ (γ non-empty)	$L = \{w \mid w$ describes a terminating Turing machine$\}$

* Meaning of symbols:
- a = terminal
- A, B = non-terminal
- α, β, γ = string of terminals and/or non-terminals

I. **IMPORTANTE**

O que não pode ser considerado uma linguagem regular?

Linguagens que requerem a atuação de uma memória para estruturar os elementos de suas cadeias, isto é, quando a frequência de um elemento da cadeia determina a frequência de outro elemento da mesma cadeia. Portanto, linguagens como "*todas cadeias de 'a' seguido de 'b', onde o número de 'a' é igual ao de 'b'*", não são regulares pois o número de 'a' determina o número de 'b'.

Expressões Regulares

Definição de ER

Sejam R_1 e R_2 duas linguagens. Sejam também as ER r_1 e r_2 que denotam as linguagens R_1 e R_2.

Dizemos que r é uma *expressão regular* (ER) se r for:

1. $a, \forall a \in \Sigma$, ou seja, a linguagem que contém exclusivamente a palavra constituída pelo símbolo a;
2. ϵ, ou seja, a cadeia vazia;
3. $\varnothing$, ou seja, a linguagem vazia. E, indutivamente, considerando r_1 e r_2, expressões regulares, temos:
4. $r = (r_1 \cup r_2)$, a união, também denotada por $r_1 + r_2$;
5. $r = (r_1 \circ r_2)$, a concatenação, também denotada por $r_1 r_2$;
6. $r = (r_1^*)$, a operação estrela, ou concatenação sucessiva.

- Σ^*, para Σ um alfabeto qualquer
  - É a linguagem constituída de todas as cadeias de caracteres sobre Σ
  - Sendo $\Sigma = \{a, b\}, \Sigma^* = (a \cup b)^* + \epsilon$

$\Sigma = \{a, b\}$.

Expressão Regular, (R)	Linguagem gerada, $L(R)$
$ab + ba$	$L(R) = \{ab, ba\}$
$(a + b)^*$	Todas as palavras sobre Σ
$(a + b)^*aa(a + b)^*$	Todas as palavras que contém a subcadeia 'aa'
$(a + b)^*(aa + bb)$	Todas as palavras que terminam com 'aa' ou 'bb'

considere o alfabeto $\Sigma = \{a, b\}$.

ER, (R)	Linguagem gerada, $L(R)$
$(a \cup \epsilon)b^*$	$= \{w \mid w = ab^* \cup b^*\}$
$(a \cup \epsilon)(b \cup \epsilon)$	$= \{\epsilon, a, b, ab\}$
$b^*\varnothing$	$= \varnothing$
$\varnothing^*$	$= \{\epsilon\}$, (ou seja, juntar qualquer número de cadeias, pode ser pode ser juntar 0 cadeias, o que gera a cadeia vazia)

Autômatos Finitos e Gramáticas Regulares

Teorema: A toda gramática regular há um autômato finito equivalente.
Demonstra-se através da construção de um modelo de autômato a partir dos não-terminais da gramática.

Teorema: A todo autômato finito há uma gramática regular equivalente.
Demonstra-se através da construção de um modelo de uma gramática cujos não-terminais são os estados do autômato.

Equivalência entre autômato finito determinístico (AFD) e autômato finito não-determinístico (AFND)

Teorema: Para cada modelo de autômato não-determinístico há um modelo de autômato determinístico equivalente.

EXERCÍCIOS

>>> Todos os autômatos deverão ser representados usando diagrama de estados <<

1) Considere a expressão regular $\mathbf{b^*a(b^*ab^*ab^*)^*}$ sobre o alfabeto $\Sigma = \{a, b\}$ e especifique 3 palavras que pertencem e 3 palavras que não pertencem a linguagem correspondente.
Palavras que pertencem à linguagem (número ímpar de 'a's):
a: O b* inicial é vazio, e a parte (b*ab*ab*)* se repete zero vezes.
baaab: Começa com b, seguido de a. A parte (b*ab*ab*)* se repete uma vez como ab*ab*
bababa: Começa com b, seguido de a. A parte (b*ab*ab*)* se repete uma vez como b*ab*ab*.
Palavras que não pertencem à linguagem
b: Não possui o 'a' obrigatório.
aa: Tem um número par de 'a's.
bbaabb: Tem um número par de 'a's.

2) Considere o alfabeto $\Sigma = \{a, b\}$ e a linguagem $L = \{a^n bbbc^m dd \mid m, n \in \mathbb{N}_0\}$.
$\mathbb{N}_0$ significa: Conjunto dos números naturais incluindo o zero.

Existe uma expressão regular que represente essa linguagem?

Expressão regular que representa a linguagem L é: a*bbb c*dd.

3) Considere a linguagem L = { palavras que terminam com 11 } que possui como alfabeto $\Sigma = \{0, 1\}$.

a. Criar um AFD que a represente.

q0 (Estado Inicial): receber um 0, continua em q0. Se receber um 1, vai para q1, pois pode ser o início da sequência 11.

q1: Se receber um 0, volta para q0, pois a sequência foi quebrada. Se receber um 1, vai para q2, completando a sequência 11.

q2 (Estado Final): A palavra termina em 11. Se receber um 0, volta para q0. Se receber outro 1, permanece em q2, pois a nova palavra ainda termina com 11.

b. Escolha apenas UMA afirmativa que contém a expressão regular referente a L.

a) 0*1(0*1)*11*(0*1(0*1)*11*)
b) (0*1(0*1)*11*)*
c) (0*1*)*11
d) (0+1)*11*
e) (0*1(0*1)*11*)

Alternativa **c)** `(0*1*)*11`.

4) Construa uma gramática regular que gere palavras pertencentes a linguagem $L = 0^*(0+1^+)$ e depois crie um autômato finito que reconheça tal linguagem.

S → 0S
S → 0
S → 1A
A → 1A
A → 1

G = ({S, A}, {0, 1}, P, S)

P:

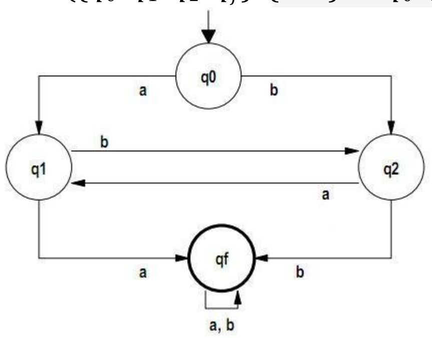
1. S → 0S
2. S → 0
3. S → 1A
4. A → 1A
5. A → 1

q0 (inicial,nao final) , q1 (final)

q0 -0,1->q1

q1-0,1->q1

5) Considere o autômato (M) e a linguagem (L) abaixo.
$M = (\{q_0, q_1, q_2, q_f\}, \{a, b\}, \delta, q_0, \{q_f\})$



$L = \{ w \mid w \text{ possui } aa \text{ ou } bb \text{ como subpalavra}\}$

Usando Função de Transição Estendida (δ*) ou alternativa apresentada pelo professor, mostre o processamento das palavras: (a) **babab** e (b) **abaab**, indicando se seriam reconhecidas ou não.
Dica: Veja no AVA sobre AFD -> Slide 11 e conteúdo presente no tópico.

Autômato (M):

- **Estados: {q0, q1, q2, qf}**
- **Alfabeto: {a, b}**
- **Estado inicial: q0**
- **Estado final: {qf}**

Transições (com base no diagrama):

- **δ(q0, a) = q1**
- **δ(q0, b) = q2**
- **δ(q1, b) = q2**
- **δ(q1, a) = qf**
- **δ(q2, a) = q1**
- **δ(q2, b) = qf**

- δ(qf, a) = qf
- δ(qf, b) = qf

Processamento da palavra babab:

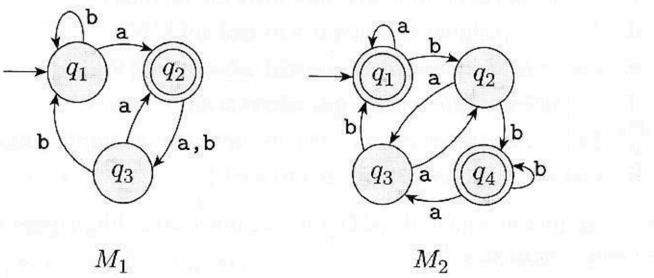
1. δ*(q0, babab)
2. δ*(δ(q0, b), abab) = δ*(q2, abab)
3. δ*(δ(q2, a), bab) = δ*(q1, bab)
4. δ*(δ(q1, b), ab) = δ*(q2, ab)
5. δ*(δ(q2, a), b) = δ*(q1, b)
6. δ*(δ(q1, b)) = q2

Processamento da palavra abaab:

1. δ*(q0, abaab)
2. δ*(δ(q0, a), baab) = δ*(q1, baab)
3. δ*(δ(q1, b), aab) = δ*(q2, aab)
4. δ*(δ(q2, a), ab) = δ*(q1, ab)
5. δ*(δ(q1, a), b) = δ*(qf, b)
6. δ*(δ(qf, b)) = qf

---

6) A seguir estão os diagramas de estado de dois AFDs (M_1 e M_2).
Responda às seguintes questões sobre cada uma dessas máquinas.



a. Qual é o estado inicial?
b. Qual é o conjunto de estados de aceitação?
c. Por qual seqüência de estados a máquina passa para a entrada aabb?
d. A máquina aceita a cadeia aabb?
e. A máquina aceita a cadeia ε?

a)q1 (indicado pela seta sem origem).
b){q2} (indicado pelo círculo duplo).
c) Sequência: **q1, q2, q2, q3, q3**
d)Não, pois o processamento termina no estado q3, que não é de aceitação.
e) Não, pois o estado inicial q1 não é de aceitação.
Máquina M2:
a)q1 (indicado pela seta sem origem).
b){q1, q4} (indicados pelos círculos duplos).
c) Sequência: q1, q2, q3, q4, q4
d) Sim, pois o processamento termina no estado q4, que é um estado de aceitação.
e) Sim, pois o estado inicial q1 também é um estado de aceitação.

---

7) Converta o AFN-ε (Autômato Finito Não Determinístico Vazio) para AFD:

Estados: {q0, q1, q2}

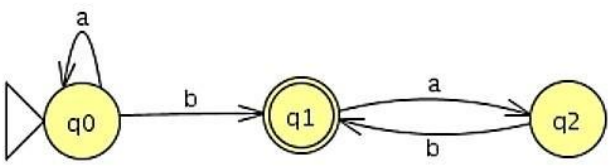
Alfabeto: {a, b, c}

Estado Inicial: q0

Estado Final: {q2}

A	B	C	D
Estado Atual	Com 'a'	Com 'b'	Com 'c'
->*A	A	B	C
*B	D	B	C
*C	D	D	C
D	D	D	D

8) Construa uma gramática regular que gera a linguagem reconhecida pelo AFD abaixo:



Estados: {q0, q1, q2}
Símbolo Inicial: q0 (que corresponderá ao símbolo inicial da nossa gramática)
Estado Final: {q1}
Transições:

- δ(q0, a) = q0
- δ(q0, b) = q1
- δ(q1, a) = q2
- δ(q2, b) = q1

G = ({q0, q1, q2}, {a, b}, P, q0)

P (Produções):

- q0 → a q0 | b q1 | b
- q1 → a q2
- q2 → b q1 | b

Esta gramática gera exatamente a mesma linguagem que o AFD reconhece, que pode ser descrita pela expressão regular a*b(ab)*.