"palavra vazia" OU "1 ocorrência de a" OU "2 ou mais ocorrências de a"

INF 1022 – P1 de Anal. Sintáticos e Léxicos – 2021.1
Prof Edward Hermann Haeusler

Daniel Huf
1920468

Resolva as questões abaixo.

1. Apresente expressões regulares que descrevam as linguagens abaixo:

   (a) Linguagem das palavras sobre $\{a, b, c\}$ tal que entre duas ocorrências de a existe pelo menos uma ocorrência de b.

   $$\varepsilon + (b+c)^*a(b+c)^* + (b+c)^*(a(b+c)^*b(b+c)^*)^*(b+c)^*$$

   (b) Linguagem das palavras sobre $\{a, b, c, d\}$ onde o primeiro símbolo na palavra não ocorre no meio da mesma e o último símbolo é diferente do primeiro.

   ≈ aceita palavras com só 1 símbolo

   $$(\varepsilon + a(b+c+d)^*(b+c+d) + b(a+c+d)^*(a+c+d) + c(a+b+d)^*(a+b+d) + d(a+b+c)^*(a+b+c))$$

   (c) Linguagem das palavras sobre $\{a, b, c\}$ que têm exatamente 3 ocorrências de a, ou não têm nenhuma ocorrência de c, ou se iniciadas em abc terminam em bca.

   $$(b+c)^*a(b+c)^*a(b+c)^*a(b+c)^* + (a+b)^* + abc(a+b+c)^*bca + (b+c+aa+ac+aba+abb)(a+b+c)^*)$$

2. O autômato a seguir representa a linguagem das palavras sobre $\{a, b\}$ nas quais há quantidade ímpar de a e par de b.

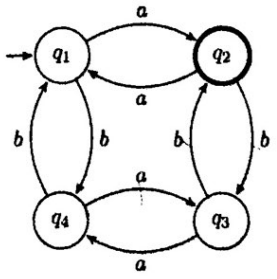
   (a) Ele é determinístico? Justifique.

   (b) Como você pode alterá-lo para que ele aceite a linguagem das palavras nas quais há quantidade ímpar de a **ou** par de b, para o mesmo alfabeto?

   (c) Com base na sua resposta anterior, atribua um ***significado*** a cada estado desse autômato. Em outras palavras: quando estamos em um dos estados do autômato, o que garantimos sobre o que já foi lido da palavra recebida de entrada? Dica: faça alguns testes com palavras de entrada para pegar essa intuição.

(Nota): Informalmente: "1 a ocorre 3 vezes" OU "c não ocorre" OU "abc termina em bca" OU "¬ abc termina em qualquer coisa"

ε
b
c
aa
ac
aba
abb

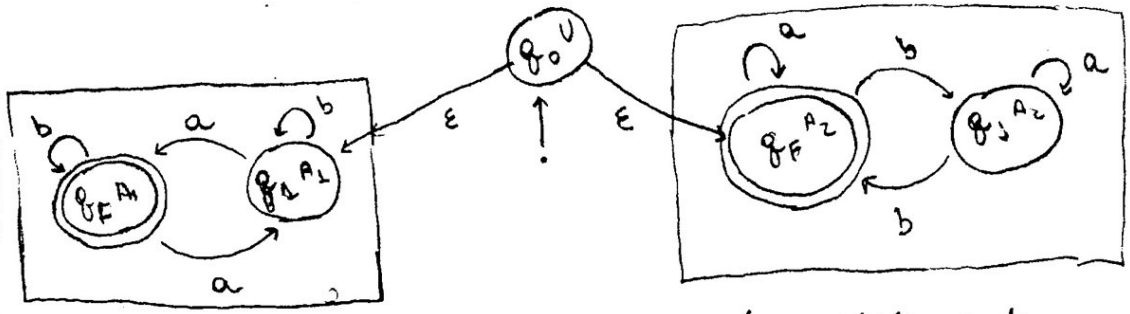


(a) Sim, pois trata-se de um autômato em que o reconhecimento de uma transição de um estado para outro é determinado pela entrada, de forma que o processamento é único. Em outras palavras, uma entrada corresponde à passagem de um estado para um ÚNICO outro (ou o mesmo) estado.

(b) Pelas propriedades da linguagem de uma expressão regular (que pode ser descrita por um AFD ou AFND), temos que $L(e_1+e_2) = L(e_1) \cup L(e_2)$. Assim podemos transformar a linguagem em questão em uma união de linguagens e construir um autômato com transições ε.

↳ CONTINUA NA PRÓXIMA PÁGINA

(b) Assim, temos:



(quantidade ímpar de a's e arbitrária de b's)

(ou)

(quantidade par de b's e arbitrária de a's)

(c) Quando estamos no estado q_0^U, através da transição ε já passamos automaticamente para um dos estados iniciais de cada uma das linguagens que estão em união. Para a linguagem à esquerda, o primeiro estado representa a palavra vazia inicialmente. Na passagem de $q_1^{A_1}$ para $q_F^{A_1}$, temos um número ímpar de a's, o que já pode ser aceito pelo autômato. Na passagem de $q_F^{A_1}$ para $q_1^{A_1}$, temos um número par de a's, de modo que para atingirmos um estado de aceitação, devemos voltar para $q_F^{A_1}$, tornando a quantidade de a's ímpar novamente. Nos dois estados, podemos incluir (ou não) b's livremente, já que sua quantidade é arbitrária para tal linguagem. Para a linguagem à direita, o estado $q_F^{A_2}$ já aceita inicialmente a palavra vazia, já que possui quantidade par de b's (0). Passando de $q_F^{A_2}$ para $q_1^{A_2}$, a string passará a ter quantidade ímpar de b's, devendo retornar para $q_F^{A_2}$ para ser aceita, já que é adicionado mais um b. Novamente, em ambos os estados podemos incluir (ou não) a's livremente, já que sua quantidade é arbitrária para tal linguagem.

3. Em aula vimos em detalhes dois métodos para a conversão de Autômatos Finitos para ERs: o método de sistemas de equações (baseado no lema de Arden) e o método recursivo. Monte o sistema de equações para o autômato do **enunciado** anterior (não é necessário resolvê-las), e argumente:
   (a) Como seriam realizadas as etapas seguintes desse processo.
   (b) Como funciona o método recursivo quando aplicado a esse autômato.

→ Sistema de equações:

$L_{q_1} = \varepsilon + L_{q_2}a + L_{q_4}b$

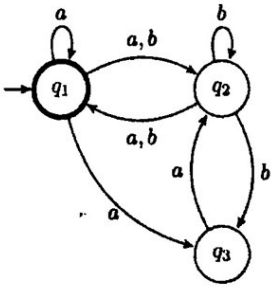
$L_{q_2} = L_{q_1}a + L_{q_3}b$

$L_{q_3} = L_{q_2}b + L_{q_4}a$

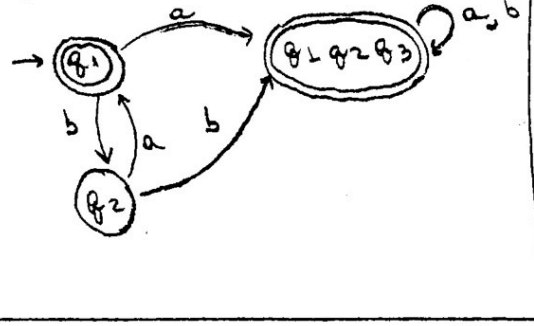
$L_{q_4} = L_{q_1}b + L_{q_3}a$

(a) O objetivo de realizar o sistema é encontrar um formato para L_{q_2} (estado de aceitação) que descreva a linguagem do autômato através de uma ER. Para isso, deve-se fazer substituições de modo que consigamos obter uma expressão no formato $X = AX + B$ ou $X = XA + B$, aplicando o lema de Arden à direita ou à esquerda respectivamente de modo a simplificar o sistema de equações. No final, devemos chegar a uma representação de L_{q_2} (X = A*B e X = BA*) que esteja somente em função de a's e b's. (b) NA PRÓXIMA FOLHA

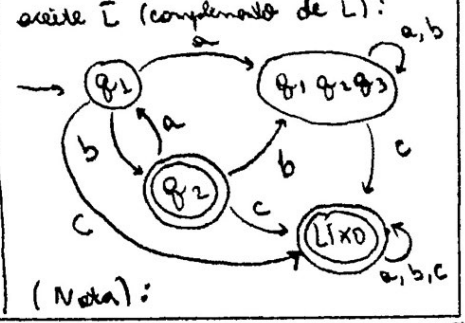
4. Seja o Autômato Finito abaixo que aceita a linguagem L sobre o alfabeto $\Sigma = \{a, b, c\}$. Apresente um Autômato Finito Determinístico que aceite a linguagem $\overline{L} = \Sigma^* - L$. Dica: primeiro torne-o determinístico.



→ Tornando o AFND em AFD:



→ Criando um AFD que aceite $\overline{L}$ (complemento de L):



(Nota):

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

$A(\overline{L}) = (Q, \Sigma, q_0, Q - F, \delta)$

Para a construção de um autômato que aceite $\overline{L}$, basta trocar os estados de aceitação de L para não aceitação, e vice-versa, e adicionar um "lixo", que é para onde vai o resto das linguagens aceitas

3. (b) O método recursivo pode ser descrito através do método de decomposição do autômato. Pelo método, α_{uv}^{X} pode ser entendido como a linguagem das palavras associadas a um caminho no autômato que começa em u, passa somente por estados em X, e chega a v. Para a questão, teremos $\alpha_{q_1 q_2}^{Q}$, onde Q representa todos os estados do autômato, ou seja, $\{q_1, q_2, q_3, q_4\}$. Assim, teremos:

① Caso Base: (poderia ser outro caso base)

$$\alpha_{q_1,q_2}^{\emptyset} = \sum_{\sigma \in \{\sigma : \delta(q_1,\sigma) = q_2\}} = a$$

② Passo recursivo:

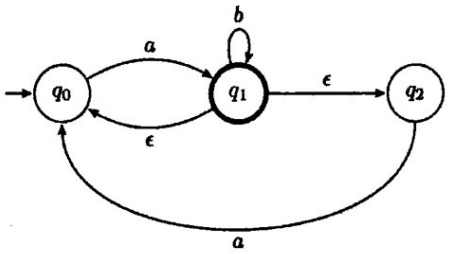
$$\alpha_{q_1,q_2}^{Q} = \alpha_{q_1,q_2}^{Q-q} + \alpha_{q_1 q}^{Q-q} \cdot (\alpha_{qq}^{Q-q})^{*} \cdot \alpha_{qq_2}^{Q-q}$$

→ Para algum $q_i \in Q$, $Q - q$ representa um conjunto de estados do autômato que não inclui um estado q do autômato.

→ Deve-se fazer a soma de todas as possibilidades de q_i, ou seja, q_1, q_2, q_3, q_4.

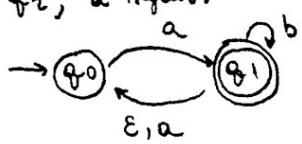
→ α_{q_1,q_2}^{Q}, quando calculado recursivamente, retorna a expressão regular que descreve o autômato, já que inclui os caminhos do estado inicial ao final que não passam por algum q e os caminhos que passam por q precedidos por caminhos de q_1 a q sem passar por q (no lema de Kleene) e sucedidos por caminhos de q a q_2 sem passar por q. Fazendo todas as possibilidades de q's e após simplificação, chega-se à linguagem do autômato.

5. Considere o autômato A abaixo:



Apresente um Autômato Finito (determinístico ou não) equivalente a A e sem transições ϵ. Justifique sua resposta.

→ O objetivo é eliminar transições de estados que não consomem caracteres de entrada. Primeiramente, irei eliminar a transição ε entre q_1 e q_2, a seguir:

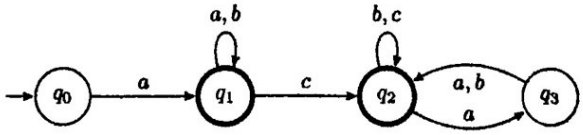


} O que eu fiz foi ligar diretamente q_1 a q_0 através da entrada a, eliminando q_2.

→ Depois, eliminei a última transição ε:

} O que "↻ ε,a" fazia era simplesmente fazer com que q_1 retornasse para q_1 com a chamada do input a.

6. Seja o AFD $A = \langle \{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \delta, q_0, \{q_1, q_2\} \rangle$, onde δ :



Argumente porque nenhum estado possui equivalente (a não ser a si mesmo) no autômato, isto é, o autômato é mínimo.

→ Analisando todas as combinações par a par de estados:
(tendo em mente que $p \approx q \iff \forall w \in \Sigma\ (\hat{\delta}(p,w) \in F \iff \hat{\delta}(q,w) \in F)$)

① Naturalmente, $q_0 \not\approx q_1$, $q_0 \not\approx q_2$, $q_3 \not\approx q_1$, $q_3 \not\approx q_2$, pois estamos comparando estados finais com estados não finais.

② Temos que:
$\hat{\delta}(q_1, a) = q_1$
$\hat{\delta}(q_2, a) = q_3$
} $q_1 \not\approx q_3 \Rightarrow q_1 \not\approx q_2$

③ Também temos que:
$\hat{\delta}(q_0, a) = q_1$
$\hat{\delta}(q_3, a) = q_2$
} $q_1 \not\approx q_2 \Rightarrow q_0 \not\approx q_3$

→ Para todas as combinações possíveis, não há estados equivalentes. //

7. Considere os pares de linguagens (expressas por ER) em cada linha da tabela abaixo.

$0^*1(01^*0+10^*1)^*$	$\supsetneq$	$(0+101^*01)^*1$
$(0+1)^*000(0+1)^*$	$\supsetneq$	$(1^*00+1^*01(0+1)^*00)0(0+1)^*$
$1(0+10^*1)^*$	$=$	$1(0+10^*1)^*(0+10^*1)^*$
$(10^*+1000)^*$	$=$	$(10^*+1010)^*$

Em cada linha da tabela, no campo central, preencha com:

- $\subsetneq$, caso a primeira linguagem seja menos expressiva que a segunda (portanto, um subconjunto),
- $\supsetneq$, caso a segunda linguagem seja menos expressiva que a primeira (portanto, um subconjunto),
- $=$, caso sejam a mesma linguagem, ou
- $\times$, caso sejam linguagens diferentes, e uma não seja subconjunto da outra.

Justifique sua resposta, exibindo contra-exemplos onde houver diferença entre as linguagens, ou argumentando, no caso de igualdade.

① A linguagem da direita é subconjunto da linguagem da esquerda, mas não é igual. A linguagem da esquerda aceita 100 por exemplo, que não é aceita pela linguagem da direita. Por outro lado, todas as palavras formadas pela linguagem da direita também podem ser formadas pela linguagem da esquerda, isso porque todas as tentativas de reproduzir as sequências 0 ou 101^*01 de 0 a ∞ vezes, seguido de 1, sempre resultará em uma string capaz de ser reproduzida pela primeira linguagem

② Na 2ª linguagem descrita pela ER, pode-se adaptá-la para que possua o prefixo $000(0+1)^*$, exatamente igual ao prefixo da 1ª linguagem. Porém, a 1ª linguagem possui o prefixo $(0+1)^*$, que equivale a todas as possibilidades de string do alfabeto Σ, ou seja, Σ^*. Em contrapartida, o prefixo utilizado da 2ª linguagem, ($1^*+1^*01(0+1)$), é menos expressivo do que $(0+1)^*$, ele não aceita 0, por exemplo.

CONTINUA NA PRÓXIMA FOLHA

8. Mostre que a linguagem $\{(^n)^n : n \geq 0\}$ (uma versão simples de balanceamento de parênteses) não é regular via lema do bombeamento. Caso ela fosse uma linguagem que atendesse ao lema, o que poderia ser dito sobre ela?

→ Prova:

① Suponha que tal linguagem seja regular, e a chamaremos de L

② Pelo lema do bombeamento: $\exists K \in \mathbb{N}, K > 0$, tal que $\forall w, w \in L, |w| > K$, $\exists uvz, |uv| \leq K, v \neq \varepsilon, \forall i \geq 0, uv^iz \in L$

③ Considere $(^K)^K$. Por hipótese, a parte a ser bombeada deve estar contida nos K primeiros símbolos da string, e V tem pelo menos um a, porque $v \neq \varepsilon$. Isso implica que $uv^2w = a^{K+|v|}b^K$, tomando $i = 2$.

④ uv^2w não pertence a esta linguagem. Chegamos a um absurdo, de modo que tal linguagem não é regular.

Boa Prova!

→ O lema do bombeamento pode ser utilizado para determinar se uma linguagem NÃO é de determinada classe de linguagem (como foi no caso acima). Porém, satisfazer o lema do bombeamento é uma condição necessária, mas não suficiente, para se fazer parte da classe, logo nada podemos dizer quanto ao pertencimento da linguagem a uma dada classe (regulares, livre de contexto etc).

7. (3) Pelas propriedades das ER's, temos que uma string qualquer w, que com o fecho de Kleene, é representada como w^*, é equivalente a uma concatenação infinita de strings w^*, ou seja, $w^* = w^*w^* = w^*w^*w^* \ldots$
Para o caso em questão, temos que $(0+10^*1)^* = (0+10^*1)^*(0+10^*1)^*$, e como o prefixo 1 também é igual para ambas as linguagens, elas são iguais.

(4) Temos que, na primeira linguagem, $(10^*)^*$ já contem a string $(1000)^*$ e na segunda linguagem, $(10^*)^*$ já contem a string $(1010)^*$, logo vemos que ambas as linguagens são equivalentes a $(10^*)^*$, e portanto são iguais.