MAC 414 – Autômatos, Computabilidade e Complexidade
Exercícios
26/10/2020 — 0:21:40

Ao provar algo, use livremente os resultados vistos em aula ou em exercícios. Ao apresentar um exemplo, **mostre** que ele tem a propriedade que você quer exemplificar.

Com relação à bibliografia, a notação e terminologia nem sempre bate com a do curso, faz parte dos exercícios entender a notação do livro, mas escrever na do curso.

L&P é o livro [1], **Sipser** é o livro [5] (original), **Rich** é o livro [8]. Se o exercício aparece em **negrito** é porque é especialmente interessante.

Muitas dessas questões foram retiradas de listas de edições anteriores do curso, e o texto pode parecer peculiar. Assim, podem haver repetições, e a ordem dos exercícios não correspondem à sequência da matéria no curso. Alguns podem ter sido feitos em aula.

A) Autômatos

1. Sejam $A, B \subseteq \Sigma^*$, e considere a equação: $X = AX + B$, na linguagem incógnita X. Seja $L = A^*B$. Mostre:

   (a) Toda solução da equação contem L.
   Sug.: mostre que $A^nB \subseteq X$, para todo $n \geq 0$.

   (b) L é uma solução da equação.

   (c) Se $\lambda \notin A$, então L é a única solução da equação.
   Sug.: suponha que $X \neq L$ é uma solução. Como $L \subseteq X$, deve existir alguma palavra $x \in X - L$. Escolha uma tal palavra, de comprimento mínimo; como X é solução da equação, x deve ter uma fatoração de um certo tipo. Use isso para chegar a uma contradição.

   (d) Dê um exemplo de equação $X = AX + B$ que tenha infinitas soluções distintas.

2. Encontre uma expressão regular para cada linguagem abaixo sobre o alfabeto $\Sigma = \{a, b\}$:

   (a) $\{x \in \Sigma^* \mid |x|_a \quad \text{é par, ou} \quad |x|_b \quad \text{é par}\}$.

   (b) $\{x \in \Sigma^* \mid x \quad \text{não contem o fator} \quad aa\}$.

   Tente demonstrar que cada expressão é correta.

3. Para $A, B \subseteq \Sigma^*$ definimos:

$$A^{-1}B = \{t \in \Sigma^* | \text{existe } s \in A \text{ tal que } st \in B\}$$
$$BA^{-1} = \{t \in \Sigma^* | \text{existe } s \in A \text{ tal que } ts \in B\}$$

   Mostre que:

   (a) $(AC^{-1})B^{-1} = A(BC)^{-1}$.

   (b) $(A^{-1}B)C^{-1} = A^{-1}(BC^{-1})$.

4. Substituição, uma generalização da noção de morfismo. Dado um conjunto A, denotamos por 2^A a família de todos os subconjuntos de A (alguns preferem a notação $\mathscr{P}(A)$). Assim, se Γ é um alfabeto, 2^{Γ^*} é a família de todas as linguagens sobre Γ, e é um monóide com a operação produto de linguagens. Assim, dada uma função $s : \Sigma \to 2^{\Gamma^*}$, ela se estende naturalmente a um morfismo $s : \Sigma^* \to 2^{\Gamma^*}$. Vamos estender s ainda mais, ao domínio 2^{Σ^*} por $s(L) = \cup_{x \in L} s(x)$, para cada linguagem L sobre Σ. Esse tipo de função é chamado uma *substituição*. Mostre que, dada uma tal substituição:

   (a) Para quaisquer linguagens A, B, $s(A \cup B) = s(A) \cup s(B)$, $s(AB) = s(A)s(B)$, $s(A^*) = s(A)^*$.

   (b) Se para cada letra $\sigma \in \Sigma$, $s(\sigma)$ é regular, então, para cada linguagem regular L, $s(L)$ é regular.

   (c) Se α, β são expressões regulares equivalentes, então substituindo-se cada letra por uma linguagem resultam duas expressões que ainda denotam a mesma linguagem. Assim, por exemplo:

   É mais ou menos fácil mostrar que $(aab)^+ = a(aba)^*ab$. Daí segue que para *quaisquer linguagens* A, B, $(AAB)^+ = A(ABA)^*AB$.

5. Para este exercício, estamos interessados em funções realizadas por programas, ou seja, f pode ser declarada como:
   string f(**int** n);
   Aqui, suponho que a linguagem de programação tem os tipos:
   string — palavras em Σ^*
   int — inteiros de tamanho arbitrário (qualquer número de dígitos)
   e que a sintaxe permite exprimir convenientemente concatenação, acesso a caracteres, e operações aritméticas. Fora isso , a sintaxe pode ser um misto de C, Pascal, Java, Perl, a gosto do freguês. Dizemos que f *enumera* $L \subseteq \Sigma^*$ se L é a imagem de f. Trocando em miúdos, isto quer dizer que (a) para todo $i \in \mathbb{N}$, $f(i) \in L$, e (b) para todo $x \in L$, existe i tal que $f(i) = x$. Note que f não precisa ser injetora, ou seja o mesmo x pode ser imagem de vários i.

   Mostre como, a partir de uma descrição de um autômato determinístico $\mathscr{A}$, produzir uma função que enumera $L(\mathscr{A})$. Se quiser caprichar, faça isso para um autômato não determinístico. Escreva a função no mais alto nível possível; defina outras funções (como você está acostumado a fazer quando programa) para fazer detalhes.

   Note que ninguém vai compilar ou testar seu programa. O importante é explicar como ele funciona, ou num comentário por fora ou em comentários embutidos no código.

   Parte extra: Mostre como construir uma função enumeradora para qualquer linguagem regular exceto $\emptyset$, a partir de uma expressão regular. Roteiro:

   (a) Construa listadores para as linguagens que consistem de uma letra e para a linguagem λ.

   (b) Mostre como, dadas funções enumeradoras para L_1, L_2, construir uma função enumeradora para $L_1 \cup L_2$. Acho que usando apontadores para funções fica mais fácil trabalhar. Enfim, este é facil.

(c) Mostre como, dadas funções enumeradoras para L_1, L_2, construir uma função enumeradora para $L_1 L_2$. Este é difícil, mas acho que muitos de vocês, se tentarem, fazem.

(d) Mostre como, dada uma função enumeradora para L, construir uma função enumeradora para L^*. Esta é *bem difícil*. Se alguém fizer isso bem feito e me convencer que está certo, pode me convencer a ganhar algum bônus ou dispensa de outra lista.

6. Construa um autômato sobre o alfabeto $\{0, 1\}$ que reconhece as palavras que são representações binárias de números divisíveis por 3 (com qualquer quantidade de zeros à esquerda).

   Sug.: Se $(x)_2 = n$ então $(x\sigma)_2 = 2n + \sigma$ e vale a relação
   $$(2n + \sigma) \bmod 3 = (2n \bmod 3) + \sigma \bmod 3.$$

7. Considere o autômato dado por

$$K = \{q_0, q_1, q_2, q_3\}, \Sigma = \{a, b\}, s = q_0, F = \{q_1, q_3\}$$
$$\delta(q_i, a) = q_j, \quad \text{onde} \quad j \equiv i + 1 \bmod 4$$
$$\delta(q_i, b) = q_j, \quad \text{onde} \quad j \equiv i - 1 \bmod 4$$

   (a) Desenhe seu diagrama, e use um dos métodos dados em aula para encontrar uma expressão regular para a linguagem que ele reconhece.

   (b) Prove que ele reconhece a linguagem $(a + b)(aa + ab + ba + bb)^*$.

8. Um *semiautômato* é uma quádrupla $\mathcal{S} = (K, \Sigma, \delta, s)$, cujas componentes têm o mesmo significado que para um autômato. Assim, se $F \subseteq K$, $\mathcal{A} = (\mathcal{S}, F)$ é um autômato, *derivado* de $\mathcal{S}$.

   (a) Mostre que a família de linguagens reconhecidas pelos autômatos derivados de um semiautômato $\mathcal{S}$ forma uma álgebra booleana, com as operações de conjuntos.

   (b) Dados semiautômatos $\mathcal{S}_i = (K_i, \Sigma, \delta_i, s_i), i = 1, 2$, sobre o mesmo alfabeto, seu *produto* $\mathcal{S}_1 \times \mathcal{S}_2$ é definido por:

$$K = K_1 \times K_2, \qquad s = (s_1, s_2),$$
$$\delta\left((p_1, p_2), \sigma\right) = \left(\delta_1(p_1, \sigma), \delta_2(p_2, \sigma)\right)$$

   Mostre que para toda palavra x vale que
   $$\delta\left((p_1, p_2), x\right) = \left(\delta_1(p_1, \sigma), \delta_2(p_2, x)\right).$$

   (c) Como no ítem anterior, seja $F \subseteq K_1$. Mostre que $L((\mathcal{S}_1, F)) = L((\mathcal{S}_1 \times \mathcal{S}_2, F \times K_2))$.

   (d) Juntando os três ítens anteriores, mostre que a união e a interseção de linguagens reconhecíveis é reconhecível.

   Sug.: dados dois autômatos, suas duas linguagens são reconhecidas por derivados do semiautômato produto dos semiautômatos correspondentes aos autômatos dados.

9. Suponha que, na sua linguagem orientada a objetos predileta, esteja definida a classe `Autômato`, com três métodos:

   - `inicial` — devolve o estado inicial,

- `delta` — função, com argumentos `estado, letra`, devolve um estado (implementa a função δ),
- `final` — função booleana, tem um estado como argumento, devolve 1 se e só se o argumento é um estado final.

Vamos supor que `estado` e `letra` são tipos já existentes.

10. Construa um reconhecedor genérico para esta classe, ou seja uma função booleana com dois argumentos, um autômato $\mathscr{A}$ e uma palavra x, e que devolve 1 se e só se $x \in L(\mathscr{A})$. A função deve funcionar em tempo linear em $| x |$.

11. Construa funções booleanas `Uniao`, `Inter`, ambas com três argumentos, autômatos A_1, A_2 e uma palavra x, e que funcionam como reconhecedores para $L(\mathscr{A}_1 \cup \mathscr{A}_2)$ e $L(\mathscr{A}_1 \cap \mathscr{A}_2)$, respectivamente. Use o exercício anterior para ter uma idéia como fazer isso. Existem duas possibilidades:

    (a) Construir um novo autômato para a união e um para a interseção, e aplicar o reconhecedor da primeira parte. Isso provavelmente vai exigir alguma mudança na descrição do objeto, para se poder por a mào no conjunto de estados.

    (b) Construir os reconhecedores **sem** construir os autômatos. Isso dá para fazer, de uma forma eficiente, e é a forma certa de se entender o produto direto: um processamento paralelo nos dois autômatos.

12. Considere o autômato dado por

$$K = \{q_0, q_1, q_2, q_3\}, \Sigma = \{a, b\}, s = q_0, F = \{q_1, q_3\}$$
$$\delta(q_i, a) = q_j, \quad \text{onde} \quad j \equiv i + 1 \bmod 4$$
$$\delta(q_i, b) = q_j, \quad \text{onde} \quad j \equiv i - 1 \bmod 4$$

    (a) Desenhe seu diagrama, e use um dos métodos dados em aula para encontrar uma expressão regular para a linguagem que ele reconhece.

    (b) Prove que ele reconhece a linguagem $(a + b)(aa + ab + ba + bb)^*$.

13. Descreva algoritmos para:

    (a) Dado um autômato determinístico, decidir se a linguagem que ele reconhece é não vazia.

    (b) Dadas duas expressões regulares, decidir se elas descrevem a mesma linguagem.

    Note bem: Na descrição dos algoritmos acima, use livremente os algoritmos vistos em aula, bem como algoritmos bem conhecidos sobre grafos, etc. Por exemplo, um algoritmo para descobrir se um autômato não determinístico aceita uma palavra da forma a^n seria: *faça uma busca em profundidade no grafo do autômato a partir dos estados iniciais, usando só arestas com rótulo em a^*; responda sim se e só se algum estado final for atingido.*

    Os algoritmos têm que estar corretos, mas não precisam ser eficientes.

14. Seja Σ um alfabeto, e sejam $A, B \subseteq \Sigma^*$. Defina

$$L_{A,B} = \{x \in \Sigma^* | \text{ nenhuma palavra de } A \text{ é segmento de } x \text{ e toda palavra de } B \text{ é segmento de } x\}.$$

Mostre que se A é regular e B é finita, então $L_{A,B}$ é regular. (Obs: esta questão tem uma solução muito simples - baseada na teoria, claro.)

15. Lembre que uma *subpalavra* de uma palavra x é qualquer palavra que se pode obter de x apagando-se algumas letras. Mais formalmente, y é subpalavra de x se existe uma fatoração $x = w_0 y_1 w_1 y_2 \dots y_n w_n$ tal que $y = y_1 y_2 \dots y_n$. Dada uma linguagem L, definimos a linguagem

$$L^\dagger = \{x \in \Sigma^* \mid \text{alguma subpalavra de } x \text{ está em } L\}.$$

Por exemplo, se $\lambda \in L$, então $L^\dagger = \Sigma^*$. Mostre como modificar um autômato não determinístico $\mathscr{A}$ obtendo outro que reconhece $L(\mathscr{A})^\dagger$.

16. Considere autômatos determinísticos $\mathscr{A} = (Q, \Sigma, \delta, s, F), \mathscr{A}' = (Q', \Sigma, \delta', s', F')$. Uma função $f : Q \to Q'$ é uma *simulação* se:

$f(s) = s'$ e

para todo $\sigma \in \Sigma, q \in Q$, $f(\delta(q, \sigma)) = \delta'(f(q), \sigma)$.

(a) Mostre que para toda palavra $x \in \Sigma^*$, $f(\delta(s, x)) = \delta'(s', x)$.

(b) A simulação é *exata* se vale: $f(q) \in F' \Leftrightarrow q \in F$. Mostre que quando existe uma simulação é exata, $L(\mathscr{A}) = L(\mathscr{A}')$.

17. Dê exemplos de linguagens não-vazias A, B e C sobre $\Sigma = \{a, b\}$ tais que

(a) $AB = AC$, mas $B \neq C$

(b) $A(B \cap C) \neq AB \cap AC$

Uma das inclusões $A(B \cap C) \subseteq AB \cap AC$ ou $AB \cap AC \subseteq A(B \cap C)$ é sempre válida. Prove-a.

18. Seja $L = \left\{x \in \{0,1\}^* : |x|_0 \neq |x|_1\right\}$. Prove que

(a) $L^* = \{0,1\}^*$

(b) $\overline{L}^* = \overline{L}$. Obs.: $\overline{L} = \Sigma^* \backslash L$, é o complemento de L.

19. Sejam A, B e C linguagens sobre Σ. Prove que:

(a) Se $A \subseteq B$ então $AC \subseteq BC$ e $CA \subseteq CB$. Dê infinitos exemplos de linguagens não-vazias A, B e C tais que $A \subset B$ e $AC = BC$.

(b) $(A \cup B)C = AC \cup BC$

(c) Se $A \subseteq B$ então $A^* \subseteq B^*$.

(d) Se $A \subseteq C^*$ e $B \subseteq C^*$ então $AB \subseteq C^*$.

(e) $(A \cup B)^* = (B^* A)^* B^*$

20. Se $\sigma \in \Sigma$, então, para $x \in \Sigma^*$, denota-se por $|x|_\sigma$ o número de ocorrências de σ em x; isto é, $|x|_\sigma = \left|\left\{ i \mid x_i = \sigma \right\}\right|$. Mostre:

    (a) $|xy|_\sigma = |x|_\sigma + |y|_\sigma$

    (b) $|x| = \sum_{\sigma \in \Sigma} |x|_\sigma$

21. Mostre que a linguagem sobre $\Sigma = \{a, b\}$ descrita por

$$\{x : |x|_a = |x|_b\} + (a+b)^*(aa + bb)(a+b)^*$$

    é regular; ou seja, encontre uma expressão regular para ela e prove que está correta. Por outro lado, $\{x : |x|_a = |x|_b\} + (a+b)^*(aaa + bbb)(a+b)^*$ não é regular, mas tem muito chão pela frente antes de provar isso.

22. Para um inteiro $k \geq 2$, considere o alfabeto $\Sigma_k = \{0, 1, \ldots, k-1\}$ dos dígitos usados para escrever em base k. Para $x \in \Sigma_k^*$, seja $[x]_k$ o inteiro que x representa em base k. Ou seja, se $x = x_0 x_1 \ldots x_m$ (note a indexação), então $[x]_k = \sum_{i=0}^{m} x_i k^{m-i}$. Para $n \geq 2, i \in \Sigma_k$, dê um AD para as linguagens:

    (a) $L_{k,n,i} = \{x \in \Sigma_k^* : [x]_k \equiv i \bmod n\}$. Ou seja, como decidir a classe de $[x]_k$ mod n com uma varredura dos dígitos da esquerda para a direita.

    (b) O reverso de $L_{k,n,i}$. Ou seja, como decidir a classe de $[x]_k$ mod n com uma varredura dos dígitos da direita para a esquerda.

    Obs: isso parece muito mais difícil do que é. Experimente com $n = 3, 4$, $i = 0, 1, 2$ e valores pequenos de k. Mesmo $k = 2$ já é interessante.

23. Construa autômatos determinísticos que reconheçam as seguintes linguagens sobre o alfabeto $\Sigma = \{a, b\}$, e dê uma expressão regular para cada uma delas:

    (a) Palavras em que cada a é imediatamente precedido por um b.

    (b) Palavras que contém o segmento $abab$.

    (c) Palavras em que o número de b's entre cada par de ocorrências de a é par

24. Construa o AD reduzido que reconhece

$$\left(b^* \Big((a+b)\,(ab^*a + b) \Big)^* (a+b)ab^* + b^* \right) \left(a \Big((a+b)\,b \Big)^* (a+b)a + b \right)^*$$

    Use os algoritmos vistos em aula, colocando detalhe suficiente para mostrar que você conhece os algoritmos, mas não todos os detalhes. Pode também curto-circuitar partes da construção do AND a partir da ER, mas cuidado porque isso pode levar a erro.

25. Mostre que $\{a^n b^p a^m \mid n + m \geq 100, p \text{ é primo}\}$ não é regular.

26. Mostre que a linguagem das expressões regulares sobre o alfabeto $\{a, b\}$, vista como linguagem sobre o alfabeto $\{a, b, +, ^*, (,), \emptyset\}$, não é regular.

27. Dada uma linguagem L, definem-se as linguagens $\text{Pref}(L) = \{x \mid \exists y \text{ tal que } xy \in L\}$ e $\text{Suf}(L) = \{x \mid \exists y \text{ tal que } yx \in L\}$. Mostre que se L é regular, então essas duas também o são, de dois jeitos diferentes:

    (a) Mostre como, dada uma expressão regular para L, obter ERs para $\text{Pref}(L)$ e $\text{Suf}(L)$.

    (b) Mostre como, dada um autômato para L, obter autômatos para $\text{Pref}(L)$ e $\text{Suf}(L)$.

    Essas construções devem ser diretas, sem usar o Teorema de Kleene. No caso de autômatos, não é necessário preservar determinismo.

28. Um *identificador* em C é uma palavra não vazia composta de letras (sem acento) maiúsculas ou minúsculas, dígitos, o sublinhado _ , e não pode começar por dígito. Mostre que o conjunto de identificadores é uma linguagem regular. Crie seu próprio açúcar sintático, se achar necessário.

29. Dê uma expressão regular para cada linguagem abaixo, onde $\Sigma = \{a, b\}$:

    (a) palavras que contém ambas as subpalavras aa e aba.

    (b) palavras que não contém letras consecutivas iguais.

    (c) palavras que não terminam em ba.

    (d) palavras que não contém a subpalavra bba.

    (e) palavras com no máximo uma ocorrência de aa e no máximo uma ocorrência de bb (por exemplo, $ababbabaaba$ está na linguagem).

    (f) palavras w tais que existe z tal que wz está na linguagem do item anterior.

30. Para cada linguagem da questão anterior, dê um autômato determinístico que a reconheça.

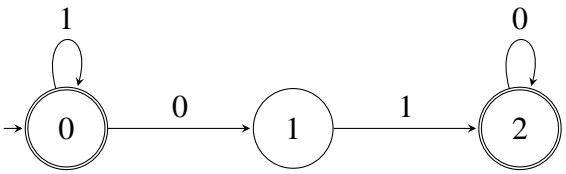
31. Uma palavra u é uma *intercalação* das palavras x e y, se existe uma fatoração $u = u_1 v_1 u_2 v_2 \cdots u_n v_n$ tal que $x = u_1 u_2 \dots u_n$ $y = v_1 v_2 \dots v_n$. Por exemplo, as intercalações de **ba** e abc são **ba**abc, **b**a**b**c, **b**abc**a**, a**ba**bc, a**b**b**a**c, ab**b**c**a**, abc**ba**. O *embaralhamento* de duas linguagens L e H é

$$L \sqcup H = \{u \in \Sigma^* \ : \ u \text{ é uma intercalação de algum } x \in L \text{ e algum } y \in H\}.$$

    Mostre que se L e H são regulares, $L \sqcup H$ também é.

32. Dadas linguagens L e H, definimos a linguagem $\frac{L}{H} = \{x \in \Sigma^* \ : \ \text{existem } u, v \in H \text{ tais que } uxv \in L\}$. Mostre que se L é regular, então, para qualquer linguagem H (pode nem ser regular), $\frac{L}{H}$ é regular.

33. Escreva uma expressão regular para o complemento de $L(\mathscr{A})$, onde $\mathscr{A}$ é o AND abaixo, e $\Sigma = \{0, 1\}$. Explique como chegou nela.



34. *Joquenpô*, também conhecido como "Pedra, Papel e Tesoura", é um jogo para dois jogadores. Uma jogada consiste de os dois jogadores exibirem um gesto que representa um dos três objetos. Se os dois jogadores fazem a mesma escolha, a jogada empatou. Senão, o vencedor da jogada é determinado pelas regras

    - Tesoura ganha de Papel
    - Papel ganha de Pedra
    - Pedra ganha de Tesoura

    Mais detalhes a uma gugada de distância (ou Wikipedia direto).

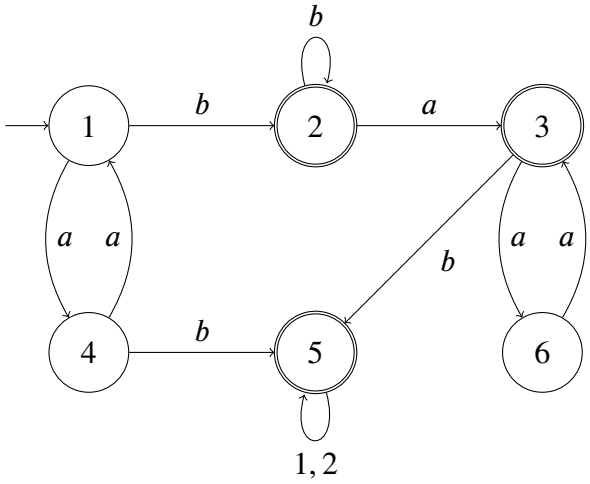
    Num torneio modelo `414` de joquenpô, os jogadores jogam sucessivamente, e o primeiro que ganhar duas jogadas vence. Seu objetivo é inventar uma codificação das jogadas tal que uma sequência delas possa ser processada por um semi-autômato que informa quem ganhou, no momento em que ganhou.

    Mais precisamente:

    - Cada letra do alfabeto deve indicar um resultado possível de jogada
    - Devem existir dois conjuntos (disjuntos) de estados finais, F_1, F_2, tais que, ao chegar em F_i significa que o jogador i ganhou.

    Os mais corajosos podem tentar descrever um semi-autômato para `414++`, em que é preciso ganhar três jogadas para vencer o torneio.

35. Para o autômato abaixo, tente descrever a linguagem que ele aceita numa frase em português matemático. Algo como "todas as palavras tais que..."



36. Considere o seguinte algoritmo, dado um AD $\mathscr{A}$:

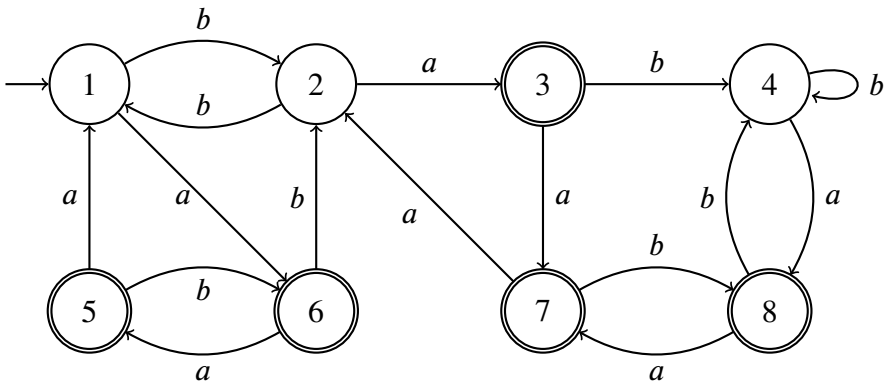
    - Faça duas vezes:
      (a) Reverta $\mathscr{A}$ (como se fosse um AND).
      (b) Aplique a construção dos subconjuntos, tomando só a parte acessível.
    - Devolva o AD resultante.

    Claro que o autômato $\mathscr{A}$' devolvido pelo algoritmo reconhece a mesma linguagem que $\mathscr{A}$.

    (a) Mostre que esse autômato tem no máximo tantos estados quanto $\mathscr{A}$.

    (b) Mostre que repetindo essa construção com $\mathscr{A}$' resulta num autômato isomorfo a $\mathscr{A}$'.

    (c) Aplique o algoritmo no autômato abaixo (devolva simplesmente a sucessão de desenhos dos autômatos que obtiver, com os estados rotulados de um modo que seja fácil de acompanhar a construção):



37. Este exercício trata de linguagens sobre alfabeto de uma letra; para fixar as ideias, $\Sigma = \{a\}$. Dada $L \subseteq \Sigma^*$, denote $E(L) = \{n \in \mathbb{N} \mid a^n \in L\}$.

    Uma *progressão aritmética (PA)* de razão r é um conjunto de inteiros da forma $P(h, r) = \{h + nr \mid n \in \mathbb{N}\}$, onde $h, r \in \mathbb{N}$. Note que $P(h, 0) = \{h\}$.

    (a) Mostre que um AD sobre Σ cujos estados sejam todos acessíveis pode ter seus estados numerados $0, 1, \ldots, n-1$, de forma que, para $i < n-1$, $\delta(i, a) = i + 1$.

    (b) Mostre que se $E(L)$ é uma PA, então L é regular.

    (c) Prove que, para $L \subseteq \Sigma^*$, são equivalentes:

    i. L é regular.

    ii. $E(L)$ é união de um conjunto finito com uma coleção de PAs com a mesma razão.

    iii. $E(L)$ é união de uma coleção finita de PAs.

38. Dê um exemplo de linguagem sobre alfabeto de uma letra que não é regular. Use o exercício anterior e encontre uma prova rigorosa.

39. **Sipser** 1.6
40. **Sipser 1.15**
41. **Sipser** 1.22
42. **Sipser 1.37**
43. **Sipser** 1.38.
44. **L&P** 1.7.5.
45. **L&P** 1.8.3.
46. **L&P** 1.8.5.
47. **L&P** 2.4.3.
48. **L&P** 2.4.5.
49. **L&P 2.4.6**.
50. **L&P 2.4.7**.
51. **L&P 2.4.8**.
52. **L&P** 2.4.13.
53. **L&P** 2.5.1.
54. **L&P** 2.6.1.
55. **L&P** 2.6.2.
56. **L&P 2.6.3**.

B) Computabilidade

1. Mostre que uma linguagem $L \subseteq \Sigma^*$ é recursivamente enumerável sse existe uma linguagem recursiva $H \subseteq \Sigma^*\#\Gamma^*$ para algum $\Gamma \supset \Sigma$ (e um símbolo especial $\# \in \Gamma \backslash \Sigma$) tal que $L = \{x \in \Sigma^* \mid \text{existe } y \in \Gamma^* \text{ tal que } x\#y \in H\}$. (Obs1: o # está aí para separar componentes de um par ordenado; em vez de $x\#y$, a definição poderia usar (x, y); Obs2: podemos pensar no y como sendo um certificado de que $x \in L$)

2. Descreva em português como funcionaria uma MT para reconhecer a linguagem $\{ww \mid w \in \Sigma^*\}$. A MT pode ter várias fitas, ser não-determinística, o que você quiser. Use uma linguagem de mais ou menos alto nível, com operações pequenas como "andar pela fita", "copiar um caractere", etc.

3. Mostre que a família das linguagens recursivas é fechada pelas operações de união, concatenação, estrela, interseção, reversão.

4. É decidível, dadas duas linguagens recursivas L_1, L_1, se $L_1 \backslash L_2$ é regular? Recursivamente enumerável?