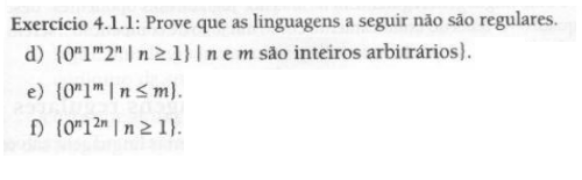
**Linguagens Formais e Autômatos**

Arthur do Prado Labaki – 11821BCC017

**Segunda Lista de Exercícios**

**Parte 2**

1) **Lema do Bombeamento para linguagens regulares:**



d) Assumindo que L é regular, o lema do bombeamento é valido.

Sendo p o valor dado pelo lema e w = 0p2p em que p >= 1.

Como |w| ≥ p, w pode ser escrito como uvz.

Qualquer que seja a divisão em que v ≥ 1 e |uv| ≤ p, temos v = 0t, para algum t ≥ 1 e u = 0r, para algum r ≥ 0.

Então w’ = uvvz em que i = 2, temos que r+2t+p-r-t = p+t. Com isso temos que w’ = 0p+t2p e como t tem seu tamanho mínimo de 1, nunca terá a mesma quantidade de 0’s e 2’s.

Logo uvvz não pertence a L, uma contradição do lema. Então L não é uma linguagem regular.

e) Assumindo que L2 é regular, o lema do bombeamento é valido.

Sendo p o valor dado pelo lema e w = 0p1p+1.

Como |w| ≥ p, w pode ser escrito como uvz.

Qualquer que seja a divisão em que v ≥ 1 e |uv| ≤ p, temos v = 0t, para algum t ≥ 1 e u = 0r, para algum r ≥ 0.

Então w’ = uvvz em que i = 2, temos que r+2t+p-r-t = p+t. Com isso temos que w’ = 0p+t1p+1 e como t tem seu tamanho mínimo de 1 ainda pode satisfazer o lema, pois n <= m.

Então w’’ = uvvvz em que i = 3, temos que r+3t+p-r-t = p+2t. Com isso temos que w’’ = 0p+2t1p+1, agora sim, o tamanho mínimo do expoente do 0 será p+2, fazendo n > m.

Logo uvvvz não pertence a L2, uma contradição do lema. Então L2 não é uma linguagem regular.

f) Assumindo que L3 é regular, o lema do bombeamento é valido.

Sendo p o valor dado pelo lema e w = 0p12p em que p >= 1.

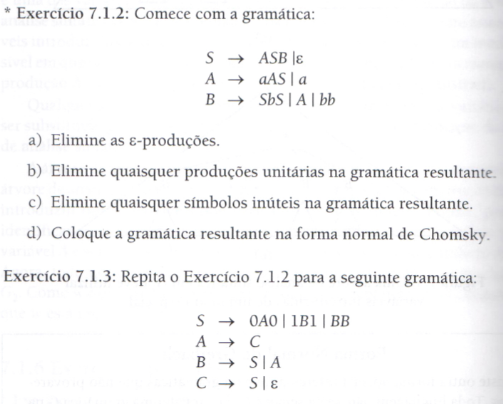
Como |w| ≥ p, w pode ser escrito como uvz.

Qualquer que seja a divisão em que v ≥ 1 e |uv| ≤ p, temos v = 0t, para algum t ≥ 1 e u = 0r, para algum r ≥ 0.

Então w’ = uvvz em que i = 2, temos que r+2t+p-r-t = p+t. Com isso temos que w’ = 0p+t12p e como t tem seu tamanho mínimo de 1, a quantidade de 1’s não será o dobro dos 0’s.

Logo uvvz não pertence a L3, uma contradição do lema. Então L3 não é uma linguagem regular.

2) **Simplificação e normalização de Gramáticas Livres de Contexto**



7.1.2)

a) S -> ASB | AB | ε, A -> aAS | aA | a, B -> SbS | b | A | bb

b) S -> ASB | AB | ε, A -> aAS | aA | a, B -> SbS | b | aAS | aA | a | bb

c) S -> ASB | AB | ε, A -> aAS | aA | a, B -> SbS | b | aAS | aA | a | bb

d) S -> XASB | AB | ε, A -> XaAS | XaA | a, B -> XSbS | b | XaAS | XaA | a | bb,

XAS -> AS, XaA -> XaA, Xa -> a, XSb -> SXb, Xb -> b

7.1.3)

a) S -> 0A0 | 00 | 1B1 | 11 | BB | ε, A -> C, B -> S | A, C -> S

b) S -> 0A0 | 00 | 1B1 | 11 | BB | ε, A -> 0A0 | 00 | 1B1 | 11 | BB | ε, B -> 0A0 | 00 | 1B1 | 11 | BB | ε, C -> 0A0 | 00 | 1B1 | 11 | BB | ε

c) S -> 0A0 | 00 | 1B1 | 11 | BB | ε, A -> 0A0 | 00 | 1B1 | 11 | BB | ε, B -> 0A0 | 00 | 1B1 | 11 | BB | ε

d) S -> X0AX0 | 00 | X1BX1 | 11 | BB | ε, A -> X0AX0 | 00 | X1BX1 | 11 | BB | ε, B -> X0AX0 | 00 | X1BX1 | 11 | BB | ε, X0A -> X0A, X0 -> 0, X1B -> X1B, X1 -> 1

3) **Ambiguidade**

Seja a gramática a seguir para expressões lógicas

G = ({S}, {∧ , ∨ , ¬ , →, a, b, c}, P, S) e

P = {S → S∧S | S∨S | S→S | ¬S | a | b | c}

a) Remova a ambiguidade considerando as seguintes precedências para os operadores lógicos: {¬ } > { → } > {∧ , ∨}.

Removendo a ambiguidade da gramatica utilizando as regras de precedências, temos:

G2 = ({S, T, F, I}, {∧ , ∨ , ¬, →, a, b, c}, P2, S) em que:

P2 = {S → S∧T | S∨T | T,

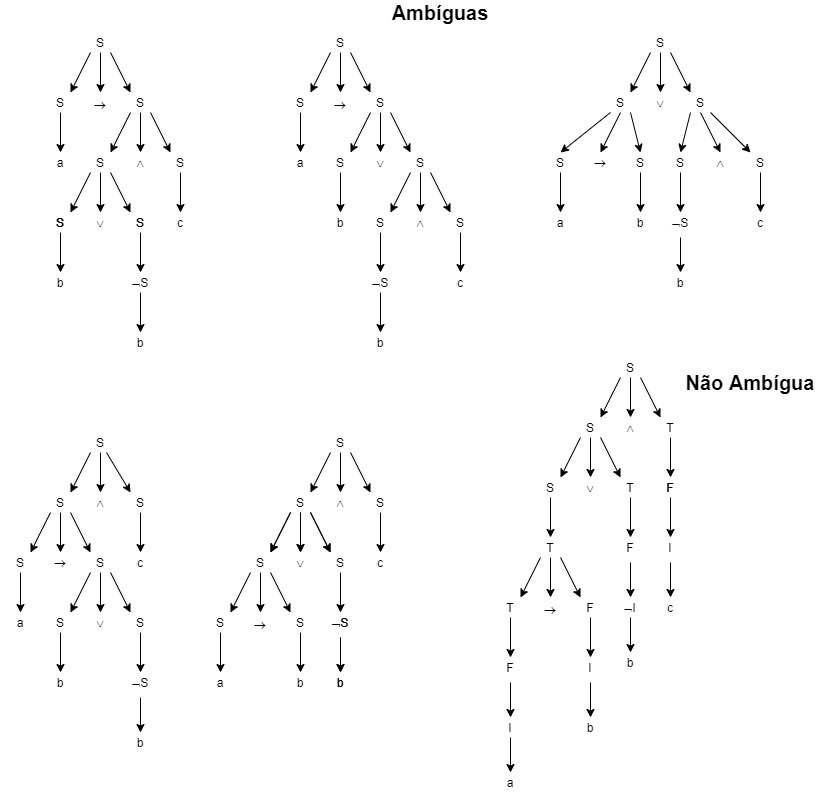
T → T → F |F,

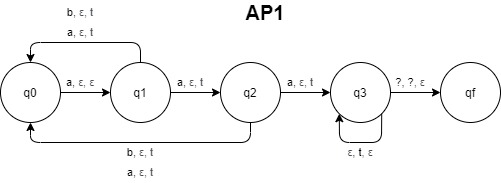
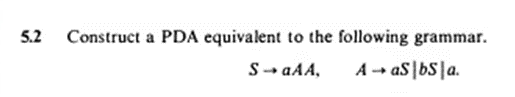
F → ¬ I | I,

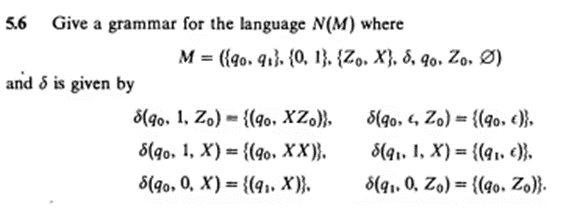
I → a | b | c}

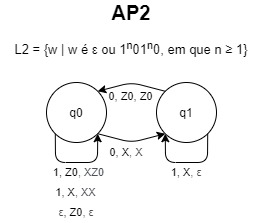
b) Mostre todas as árvores de derivação possíveis nas duas gramáticas (ambígua e não ambígua) para a expressão:

**a → b ∨ ¬b ∧ c**

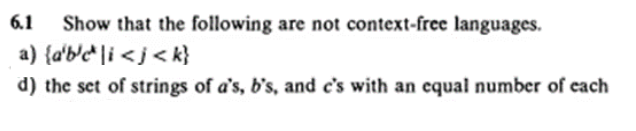


****4) **Autômato de Pilha**

****

****

5) **Lema do Bombeamento para Linguagens Livres de Contexto**

****

a) Suponha que é livre de contexto, então o lema do bombeamento vale.

Supondo que w = apbp+1cp+2, |w| ≥ p, então w = uxvyz em que:

|xvy| ≤ n, |xy| ≥ 1 e para todo i ≥ 0, uxivyiz.

Temos dois tipos de casos:

1- Se x e y tem apenas um símbolo cada:

X = ai e y = aj | bj | ε X = bi e y = bj | cj | ε

X = ci e y = cj | ε X = ε e y = aj | bj | cj

Com i, j ≥ 1. Em qualquer um desses casos, uvz terá quantidades não pertencente as regras, como ter mais a’s do que b’s.

2- Se x e y tem dois símbolos:

X = aibj e y = bk | ε X = bicj e y = ck | ε

X = ai | ε e y = ajbk X = bi | ε e y = bjck

Com i, j, k ≥ 1. Em qualquer um desses casos, uvz terá quantidade que também não pertence as regras.

Nos dois casos, não importa como w é dividida, temos que ux0vy0z não pertence a linguagem. Logo a linguagem não é livre de contexto.

d) Suponha que é livre de contexto, então o lema do bombeamento vale.

Supondo que w = apbpcp, |w| ≥ p, então w = uxvyz em que:

|xvy| ≤ n, |xy| ≥ 1 e para todo i ≥ 0, uxivyiz.

Temos dois tipos de casos:

1- Se x e y tem apenas um símbolo cada:

X = ai e y = aj | bj | ε X = bi e y = bj | cj | ε

X = ci e y = cj | ε X = ε e y = aj | bj | cj

Com i, j ≥ 1. Em qualquer um desses casos, ux2vy2z terá quantidades diferentes dos três símbolos.

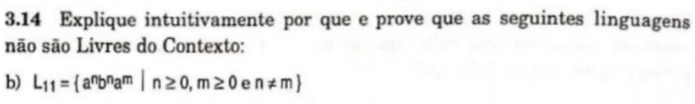
2- Se x e y tem dois símbolos:

X = aibj e y = bk | ε X = bicj e y = ck | ε

X = ai | ε e y = ajbk X = bi | ε e y = bjck

Com i, j, k ≥ 1. Em qualquer um desses casos, ux2vy2z terá os símbolos fora de ordem.

Nos dois casos, não importa como w é dividida, temos que ux2vy2z não pertence a linguagem. Logo a linguagem não é livre de contexto.

****

Suponha que é livre de contexto, então o lema do bombeamento vale.

Supondo que w = apbp, sendo m = 0, |w| ≥ p, então w = uxvyz em que:

|xvy| ≤ n, |xy| ≥ 1 e para todo i ≥ 0, uxivyiz.

Temos dois tipos de casos:

1- Se x e y tem apenas um símbolo cada:

X = ai e y = aj | bj | ε X = bi e y = bj | ε

X = ε e y = aj | bj

Com i, j ≥ 1. Em qualquer um desses casos, ux2vy2z terá quantidades diferentes dos três símbolos.

2- Se x e y tem dois símbolos:

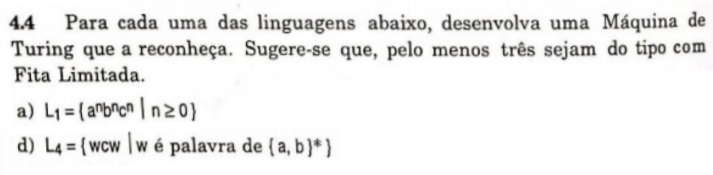
X = aibj e y = bk | ε X = ai | ε e y = ajbk

Com i, j, k ≥ 1. Em qualquer um desses casos, ux2vy2z terá os símbolos fora de ordem.

Bem semelhante ao exercício anterior, pois escolhi um w que facilita.

Nos dois casos, não importa como w é dividida, temos que ux2vy2z não pertence a linguagem. Logo a linguagem não é livre de contexto.

6) **Máquina de Turing**

****

a)

M1 = ({q0, q1, q2, q3, q4, q5}, {a, b, c}, {a, b, c, x, y, z, B}, δ, q0, {q5}) onde:

δ(q0, a) = (q1, x, →), δ(q0, B) = (q5, B, →), δ(q1, a) = (q1, a, →),

δ(q1, b) = (q2, y, →), δ(q2, b) = (q2, b, →), δ(q2, c) = (q3, z, ←),

δ(q3, b) = (q3, b, ←), δ(q3, y) = (q3, y, ←), δ(q3, a) = (q3, a, ←),

δ(q3, x) = (q0, x, →), δ(q1, y) = (q1, y, →), δ(q2, z) = (q2, z, →),

δ(q3, z) = (q3, z, ←), δ(q0, y) = (q4, y, →), δ(q4, y) = (q4, y, →),

δ(q4, z) = (q4, z, →), δ(q4, B) = (q5, B, →).

O esquema dessa MT é mudar a para x, b para y e c para z, um caractere de cada por ida na fita, e voltar até o primeiro valor a, onde irá repetir. Quando não tiver mais a (também não deve ter b’s nem c’s) a fita vai para a direita e terminara em q5. Também aceita cadeia vazia.

b)

M2 = ({q0, q1, q2, q3, q4, q5, q6, q7}, {a, b, c}, {a, b, c, x, y, B}, δ, q0, {q7})

δ(q0, a) = (q1, x, →), δ(q0, b) = (q2, y, →), δ(q0, x) = (q0, x, →),

δ(q0, y) = (q0, y, →), δ(q1, a) = (q1, a, →), δ(q1, b) = (q1, b, →),

δ(q1, c) = (q3, c, →), δ(q2, a) = (q2, a, →), δ(q2, b) = (q2, b, →),

δ(q2, c) = (q4, c, →), δ(q3, a) = (q5, x, ←), δ(q3, x) = (q3, x, →),

δ(q3, y) = (q3, y, →), δ(q4, b) = (q5, y, ←), δ(q4, x) = (q4, x, →),

δ(q4, y) = (q4, y, →), δ(q5, a) = (q5, a, ←), δ(q5, b) = (q5, b, ←),

δ(q5, x) = (q5, x, ←), δ(q5, y) = (q5, y, ←), δ(q5, c) = (q5, c, ←),

δ(q5, B) = (q0, B, →), δ(q0, c) = (q6, c, →), δ(q6, a) = (q6, a, →),

δ(q6, b) = (q6, b, →), δ(q6, x) = (q6, x, →), δ(q6, y) = (q6, y, →),

δ(q6, B) = (q7, B, →).

O esquema dessa MT é ler a, ir até c, ler o próximo a, e voltar para o começo da fita. Os dois a’s lidos são mudados para x. O mesmo acontece caso ler b, mas eles são mudados para y. Quando terminar de ler w, a máquina irá buscar o final da fita, terminando em q7.