Name: Raylander Marques Melo

Matrícula: 494563

(I) a) $L_1 = \{a^n b^m c^n d^m \mid n, m \geq 1\}$

Suponha, por absurdo, que L seja livre de contexto. Dessa forma, existe uma GLC que gera L.

Seja p o tamanho do bombeamento da linguagem L. Seja $w = a^p b^p c^p d^p$. Temos que $|w| \geq p$ e que w pertence à L. Pelo Lema do Bombeamento para linguagens livre de contexto, temos que w pode ser representado da forma uvxyz tal que $uv^i xy^i z$ pertence a L para todo $i \geq 0$. Como $|vxy| \leq p$, temos as seguintes possibilidades de bombeamento.

Dessas possibilidades vamos descartar os casos de bombear "abc", "bcd", "abcd", ou seja, no caso do "abc" se começar a bombear em "a" e terminar em "c" isso quebra a regra do lema do lema do bombeamento pois, v x y onde o v e o y é os elementos que aumentam ou diminuem eles ao aumentarem eles só podrão ter no máximo o tamanho de p, ou seja, se começar a bombear em "a" e terminar em "c" v x y terá tamanho maior do que p pois só o "b" que está entre "a" e "c" já tem o tamanho de p então ao adicionarmos a quantidade de "ab" e "cb" v x y vai passar de p. Observando que isso se aplica para "bcd" e "abcd".

Logora vou começar a mostrar as possibilidades que o lema do bombeamento permite que aconteçam.

I) Bombear apenas um tipo de símbolo:

- Tomando $i > 1$, bombeando "a" temos que a quantidade de "c's" será menor do que a quantidade de "a's", uma vez que o número de "a's" está aumentando e o número de "c's", que deveria ser a mesma quantidade de de "a's", se mantém exatamente o mesma enquanto o "a" aumenta.
- Tomando $i > 1$, bombeando "b" temos que a quantidade de "b's" será menor do que a quantidade de "b's", uma vez que o número de "b's" está aumentando e o número de "d's", que deveria ser a mesma quantidade de de "b's", se mantém exatamente a mesma, enquanto o "b" aumenta.
- Tomando $i > 1$, bombeando "c" temos que a quantidade de "a's" será menor do que a quantidade de "c's", uma vez que o número de "c's" está aumentando e o número de "a's", que deveria ser a mesma quantidade de "c's", se mantém exatamente a mesma, enquanto o "c" aumenta.
- Tomando $i > 1$, bombeando "d" temos que a quantidade de "b's" será menor do que a quantidade de "d's", uma vez que o número de "d's" está aumentando e o número de "b's", que deveria ser a mesma quantidade "d's", se mantém exatamente a mesma, enquanto o "d" aumenta.

2) Bombear dois tipos de símbolos

Podemos ter "v" ou "y" bombeando, cada um, mais de um tipo de símbolo ou apenas um único tipo de símbolo.

2.1) "v" ou "y" bombeiam mais de um tipo de símbolo.

Nesse caso, notamos a "mistura" de símbolos. Sem perda de generalidade assuma que $v = a^i b^j$, Usando f iremos bombear o v, teremos sequências de "a's" e "b's" intercaladas. Note que o resultado não possui propriedade da linguagem, pois tem que ter primeiro a sequência de "a's" e depois a sequência de "b's" nesta ordem.

2.2) v bombeia um tipo de símbolo e y bombeia outro tipo de símbolo.

2.2.1) v bombeia "a's" e y bombeia "b's"

Tomando $i > 1$, como $|vy| > 0$, temos que a quantidade de "c's" e "d's" permanece a mesma sendo que a quantidade "a's" e "b's" aumentam. Desse modo, a quantidade de "c's" não pode ser a mesma quantidade de "a's", nem a quantidade de "d's" não pode ser a mesma quantidade de "b's".

2.2.2) v bombeando "b's" e y bombeando "c's"

Tomando $i > 1$, como $|vy| > 0$, temos que a quantidade de "a's" e "d's" permanece a mesma sendo que a quantidade de "b's" e "c's" aumentam. Desse modo, a quantidade de "d's" não pode ser a mesma quantidade de "b's", nem a quantidade de "a's" não pode ser a mesma quantidade de "c's".

2.2.3) v bombeia "c's" e y bombeia "d's".

Tomando $i > 1$, como $|vy| > 0$, temos que a quantidade de "a's" e "b's" permanece a mesma sendo que a quantidade de "c's" e "d's" aumentam. Desse modo, a quantidade de "a's" não pode ser a mesma quantidade de "c's", nem a quantidade de "b's" não pode ser a mesma quantidade de "d's".

Como $uv^i xy^i z$ não pertence a L para todo i , temos que L não pode ser gerada por uma gramática livre de contexto G. Ledesvido! Assim, L não é livre do contexto.

① b) $L_2 = \{0^{i^2} 1^k \mid k = i^2\}$

Suponha, por absurdo, que L seja livre de contexto. Assim, existe uma GLC que gera L.

Seja p o tamanho do bombeamento da linguagem L. Seja $w = 0^p 1^{p^2}$.

Temos que $|w| \geq p$ e que w pertence à L. Pelo Lema do Bombeamento para linguagem livre de contexto, temos que w pode ser representado da forma uvxyz tal que $uv^i xy^i z$ pertence a L para todo $i \geq 0$. Como $|vxy| \leq p$, temos as seguintes possibilidades de bombeamento.

1) Bombear apenas um tipo de símbolo:

- Tomando $i \geq 1$, bombeando "0" temos que a quantidade de "1's" será menor do que o quadrado do número de "0's", uma vez que o número de "0's" está aumentando e o número de "1's", que deveria ser o quadrado da quantidade de "0's", se mantém exatamente a mesma, enquanto o "0" aumenta.
- Tomando $i \geq 1$, bombeando "1" temos que a quantidade de "0's" será menor, ou seja, a quantidade de "1's" não será mais apenas o quadrado da quantidade de "0's".

2) Bombear dois tipos de símbolos:

Podemos ter "v" ou "y" bombeando, cada um, mais de um tipo de símbolo ou apenas um único tipo de símbolo.

2.1) "v" ou "y" bombeiam mais de um tipo de símbolo.

Nesse caso, notamos a "mistura" de símbolos. Sem perda de generalidade assuma que $v = 0^i 1^{i^2}$. Quando formos bombear o v, teremos sequências de "0's" e "1's" intercaladas. Note que o resultado não possui propriedade da linguagem, pois tem que ter primeiro a sequência de "0's" e depois a sequência de "1's" nesta ordem.

2.2) v bombeia um tipo de símbolo e o y bombeia outro tipo de símbolo.

2.2.1) v bombeia "0's" e y bombeia "1's".

Tomando $i \geq 1$, como $|vy| > 0$, temos que a quantidade de "1's" não será o quadrado da quantidade de "0's", pois ao bombear uma vez o "0" e o "1" teremos que a quantidade deixará de ser o quadrado da quantidade de "0's", ou seja, o "1" deveria aumentar mais do que a quantidade de elementos que aumenta, fazendo com que "1" deixe de ser quadrado da quantidade de "0's". Como $uv^i xy^i z$ não pertence a L para todo $i \geq 0$, temos que L não pode ser gerada por uma gramática livre de contexto G. Absurdo! Assim, L não é livre de contexto

② S -> aAB
A -> bBb
B -> A | ε

a) abbbbb
Ler duas formas de adiquirir essa string são:

S => aAB => abBb => abAb => abbBbb => abbbbb

S => aAB => abBbA => abbbBb => abbbbb

Essa linguagem possibilita conseguir está string de duas formas, pois tem uma produção na variável "B" que possibilita executar as produções da variável "A" assim possibilitando descer e subir nas produções.

b) A gramática não ambígua criada a partir da gramática ambígua dada acima é:

S -> aAB
B -> A
A -> bAb | ε

A gramática não ambígua foi gerada percebendo que a ambiguidade só acontece as produções da variável B possibilitam subir na gramática ambígua, então sabendo disso se retirar a possibilidade de poder ir e voltar na gramática. Dessa forma agora só é possível descer na gramática não ambígua, pois ao ser chamado as variáveis "A" e "B" bem no início, ou seja, da variável "B" só tem um caminho para a variável "A" e a variável "A" agora só pode adicionar "bb" e chamar a própria variável "A" ou adicionar o vazio. Dessa forma só pode ser percorrido um caminho nesta gramática.

③ a) $L_3 = \{w \in \{a, b, c\}^* \mid N_a(w) = N_b(w) = N_c(w)\}$

$G_3 = (\{S, A, B, C\}, \{a, b, c\}, P, S)$

$P = \{$

$S \to ABCS \mid ABC$

$AB \to BA$

$AC \to CA$

$BA \to AB$

$BC \to CB$

$CB \to BC$

$CA \to AC$

$A \to a$

$B \to b$

$C \to c\}$

Essa gramática sensível ao contexto reconhece a linguagem pois, as produções da variável inicial "S" da a possibilidade de adicionar quantos "A's", "B's" e "C's" for necessário, sabendo que não importa as posições dos elementos não importa como seja adicionado os elementos. Dessa mada, temos as produções da variáveis "AB", "AC", "BA", "BC", "CB" e "CA", onde estas produções vai possibilitar trocar todos os elementos de posições, podendo assim, criar a string que desejar. Por último vem apenas as produções das variáveis "A", "B" e "C" que é onde possibilita trocar todas as variáveis por terminais.

b) $L_4 = \{a^n b^m c^{2m} \mid n, m \geq 1\}$

$G_4 = (\{S, B, a, b, c\}, \{a, b, c\}, P, S)$

P={

S -> aScc | aBcc

B -> BB

aB -> ab

Bc -> bc

BB -> bb}

Essa gramática sensível ao contexto reconhece esta linguagem, pois na variável inicial "S" tem duas produções que possibilitam adicionar "a's", "c's" e a variável B assim para cada "a" adicionado deles "c's" também são, como também, na variável "S" obriga que seja adicionado alguma coisa a string garantindo que a linguagem pede de que tenha pelo menos um elemento de "a", um de "b" e dois de "c".

Assim pode-se ser adicionado a quantidade de "b's" necessários na produção da variável "B". Dessa forma é preciso converter as variáveis "B's" existentes em terminais "b's" assim existe três produções que fazem esta conversão que é quando tem um terminal "a" seguido de uma variável "B" convertendo para os terminais "ab", também tem quando tem uma variável "B" seguida de um terminal "c" transformado em terminais "bc" e a última possibilidade de conversão é quando tem um "B" variável seguida de outro "B" variável transformando em terminais "bb".

Por último a garantia de vir os "a's" depois os "b's" e depois os "c's" é que quando os "a's" e os "c's" são adicionados os "a's" são sempre a esquerda da variável "S" e os "c's" sempre a direita ou seja, quando ao ligar do "S" entra o "B" ele só vai poder crescer no meio dos "a's" e dos "c's" assim mantendo a ordem da linguagem.

Dessa forma a gramática consegue gerar a linguagem.

⑤ Linguagem Regular:

Uma linguagem regular é uma linguagem formal que pode ser determinada por uma expressão regular, ou seja, uma linguagem produzida utilizando as operações de concatenação e união sobre elementos de um alfabeto.

As linguagens regulares são utilizadas para descrever dispositivos que realizam computações simples, como os autômatos finitos, pois representam a linguagem mais elementar classificada pela hierarquia de Chomsky que não requer memória para ser reconhecida.

Como descrição formal temos que:

- O idioma vazio é um idioma regular.
- Para cada $a \in \Sigma$, a linguagem $\{a\}$ é uma linguagem regular.
- Se A é uma linguagem regular, A^* é uma linguagem regular.
- Se A e B são linguagens regulares, então $A \cup B$ e $A \cdot B$ são linguagens regulares.
- Nenhum outro idioma acima de Σ é regular.

Exemplo de uma linguagem que pertence a linguagem regular:

$L_1 = \{a^m b^n \mid m, n \geq 0\}$

Essa linguagem pertence a uma linguagem regular pois, ela pode adicionar quantos elementos forem precisos de "a's" seguido de "b's" sem precisar contá-los como também é possível criar um autômato finito para essa linguagem.

Exemplo de uma linguagem que não pertence a linguagem regular:

$L_2 = = \{a^m b^n c^z \mid m = z\}$

Essa linguagem não pertence a uma linguagem regular pois, ela tem que possuir a mesma quantidade de "a's" e de "c's", assim é preciso que seja contado a quantidade de elementos de "a's" para que seja possível adicionar a mesma quantia de "c's". Dessa modo, a linguagem não consegue contar e nem criar um autômato finito para a linguagem.

Linguagem livre de contexto:

Uma linguagem livre de contexto é uma linguagem gerada por alguma gramática livre de contexto. Sabe-se que diferentes gramáticas livres de contexto podem gerar a mesma linguagem livre de contexto, ou, uma

dada linguagem livre de contexto pode ser gerada por diferentes gramáticas livres de contexto, precluindo assim que existe uma ambiguidade.

O conjunto de todas as linguagens livres de contexto é sintático ao conjunto de linguagens aceitas por um autômato de pilha, o que faz com que essas linguagens já consiga realizar alguns tipos de contagem. Na verdade, dada uma GLC, há uma maneira direta para produzir um autômato de pilha para a gramática como uma linguagem correspondente.

As regras de uma gramática livre de contexto são da forma que do lado esquerdo das producões ficam símbolos não terminais, e do lado direito fica uma cadeia de terminais e não terminais, ou somente terminais ou somente não terminais.

A diferença da linguagem livre de contexto para a linguagem regular é que a livre de contexto possui uma memória possibilitado pelo autômato de pilha podendo assim realizar contagem.

Exemplo de linguagem que pertence a linguagem livre de contexto:

$L_3 = \{ c^d a^n \mid d = n \}$

Essa linguagem pertence a uma linguagem livre de contexto pois, ela pode adicionar quantos elementos forem precisos de "a's" e em seguida adicionar a mesma quantidade de "b's" por ser possível contar os elementos e também é possível criar um autômato de pilha para essa linguagem que é responsável por permitir contar a quantidade de elementos.

Exemplo de uma linguagem que não pertena a linguagem livre de contexto:

$L_4 = \{ d^m f^n g^z \mid m = z \}$

Essa linguagem pertence a uma linguagem livre de contexto pois, ela pode adicionar quantos elemento forem precisos de "d's" em seguida adicionar a quantidade de "f's" que for preciso e em seguida é preciso adicionar os "g's" na mesma quantidade de "d's" porém existe uma quantidade de "f's" entre os "d's" e os "g's" assim não podendo utilizar um autômato de pilha para que seja possível ir retirando os "d's" para contar sua quantidade de adicionar os "g's" na mesma quantidade, mostrando assim que esta linguagem não é livre de contexto.

Linguagem sensível ao contexto:

Define-se inicialmente linguagem sensível ao contexto como sendo aquela que possa ser definida através de uma gramática sensível ao contexto. Assim as regras que compõem uma gramática sensível ao contexto são que:

- Tem que possuir pelo menos um não terminal do lado esquerdo.
- Precisa possuir uma sequência de símbolos do lado direito sempre maior ou igual ao lado esquerdo da mesma regra.
- Última regra é que só pode ser adicionada vazio a uma única regra e que a variável dessa regra não apareça de novo em outro lugar.

Assim temos as restrições de criação de uma gramática sensível ao contexto.

A diferença da linguagem sensível ao contexto para a linguagem livre de contexto é que na sensível ao contexto é possível ter elementos na esquerda das regras da gramática que possua não terminais e terminais juntos como também possibilita a troca de mais de um símbolo.

Exemplo de uma linguagem que pertence a linguagem sensível ao contexto:

$L_5 = \{a^n b^n c^n \mid n > 0\}$

Essa linguagem pertence a uma linguagem sensível ao contexto pois, ela pode adicionar quantos elementos forem precisos de "a's", em seguida adicionar a mesma quantidade de "b's" e em seguida adicionar a mesma quantidade de "c's" com isso possível adicionar todos ao mesmo tempo e a gramática possibilitar alterar as posições dos elementos da string deixando-os na posição que for preciso.

Exemplo de uma linguagem que não pertence a linguagem sensível ao contexto:

$L_6 = \{a^n b^m \mid m = n^2\}$

Essa linguagem não pertence a uma linguagem sensível ao contexto pois, ela pode adicionar quantos elementos forem precisos de "a's", porém ela não consegue adicionar a quantidade necessárias de "b's", pois será preciso usar ao parenciação e para isso acontecer é quebrado as regras da gramática sensível ao contexto, como por exemplo o lado esquerdo apenas terminais e isso na gramática sensível ao contexto não pode.

Linguagem recursiva

A linguagem recursiva é chamada recursiva se é um subconjunto recursivo no conjunto de todas as palavras possíveis sobre o alfabeto da linguagem. Equivalentemente, uma linguagem é recursiva se existe uma máquina de Turing que sempre para quando recebe uma sequência finita de símbolos do alfabeto da linguagem como entrada e que aceita exatamente as palavras do alfabeto da linguagem que são parte da linguagem e rejeita todas as outras palavras.

A diferença da linguagem recursiva para a linguagem sensível ao contexto é que é impossível ela a esquerda das regras da gramática seja possível apenas terminais, ou seja possível colocar vazio em qualquer lugar, como também que o lado esquerdo da regra seja maior do que o lado direito da regra, ou seja, na recursiva é possível fazer tudo.

Exemplo de uma linguagem que pertence a linguagem recursiva:

$L_7 = \{a^n b^m / m = 2^n\}$

Essa linguagem pertence a uma linguagem recursiva pois, ele pode adicionar quantos elementos forem precisos de "a's" e ir adicionando a quantidade de "b's" de acordo com a quantidade de "a's", ou seja, para cada "a" adicionado é adicionado o dobro de "b's", isso só é possível pois é possível percorrer a string.

Linguagem recursivamente enumerável:

Uma linguagem recursivamente enumerável formal é um subconjunto recursivamente enumerável no conjunto de todas as palavras possíveis sobre o alfabeto da linguagem.

Uma linguagem recursivamente enumerável é uma linguagem formal para a qual existe uma máquina de Turing que irá enumerar todas as cadeias válidas da linguagem. Note que, se a linguagem é infinita, o algoritmo de enumeração fornecido pode ser escolhido de forma que evite repetições, uma vez que podemos testar se a cadeia produzida, use a saída da entrada n+1, mas mais uma vez teste se é uma cadeia nova.

Uma linguagem recursivamente enumerável é uma linguagem formal para a qual existe uma máquina de Turing que irá parar e aceitar quando se roda com qualquer cadeia da linguagem na entrada e pode parar e rejeitar ou entrar em loop quando se roda com qualquer cadeia que não é da linguagem.

Esta é a diferença entre linguagem recursiva, que exige que a máquina de Turing sempre pare.

Exemplo de uma linguagem que pertence a linguagem recursivamente enumerável:

Como a linguagem abrange todas as outras, irei mostrar um exemplo parecido ao da linguagem recursiva.

$L_8 = \{ a^n b^m \mid m = n^2 \}$

Essa linguagem pertence a uma linguagem recursivamente enumerável pois, ela pode adicionar quantos elementos forem precisos de "a's" e ir adicionando a quantidade de "b's" de acordo com a quantidade de "a's"; ou seja, para cada "a" adicionado é adicionado uma quantidade de "b's"; isso só é possível pois, tem-se a possibilidade de percorrer a string.