Alfabeto/Linguagem

Um alfabeto (Σ) é um conjunto finito de símbolos, por exemplo {0,1}, com esse alfabeto pode-se formar Linguagens, que são concatenações com os dois símbolos do alfabeto. Um exemplo de linguagem sob o alfabeto {0,1} pode ser L = {00,01,10,11}, ou seja, os números binários de tamanho 2, uma linguagem com quatro palavras, ou seja, com quantidade finita de palavras

Uma linguagem também pode ser infinita, por exemplo, a linguagem de todos os números binários de qualquer tamanho L = {0, 1, 00, 01, 10, 11, 000, 001, ...} e assim por diante

As linguagens são representadas na forma de uma expressão que define as palavras que a compõem, por exemplo, para o alfabeto = {a}, a linguagem $L=\{a^n \ / \ n > 0\}$ é a linguagem formada por n concatenações do símbolo {a}, resultando em L = {a, aa, aaa, aaaa, ...} e assim por diante.

Outro exemplo de expressão de linguagem $L = \{a^n b^{2m} \ / \ n > 0, m > 0\}$ para o alfabeto {a,b}. Esta linguagem contém palavras que começam com pelo menos um a, podendo ter infinitos, depois dos a´s que tiverem, temos pelo menos 2 b´s, podendo ter qualquer quantidade par de b, resultando nas palavras {abb, aabb, aaabb, aaaabb, abbbb, aabbbb}, e assim por diante. Uma expressão no formato acima define ordem para todas as palavras da linguagem, ou seja, os n a's da palavra são antes dos m b´s da palavra, que estes b´s são sempre em quantidade par.

Uma linguagem também pode ser representada da seguinte forma: para um alfabeto Σ = {a,b} temos L = {w ϵ Σ*}, isto quer dizer que a Linguagem L é formada pela palavra genérica w, esta palavra w pertence a Σ* que representa todas as palavras possíveis de se formar com {a,b}, em qualquer ordem e qualquer tamanho, incluindo uma palavra especial, de tamanho zero, a palavra vazia (λ), ou seja, as palavras de L são {λ, a, b, aa, ab, ba, bb, aaa, ...}

Para o alfabeto Σ = {a,b}, mostre as palavras da seguinte linguagem, L = {w ϵ {a,b}* / w começa com a}, ou seja, L é formada por todas as palavras que começa com um a, pelo menos, e depois deste a podemos ter qualquer combinação de a e b, L = {a, aa, ab, aaa, aab, ...}

Para o alfabeto Σ = {0, 1}, mostre as palavras da seguinte linguagem L = {w ϵ {0, 1}* / |w| > 1}. Esta expressão indica que as palavras da linguagem são formadas pela palavra genérica w, que são combinações em qualquer ordem e qualquer tamanho de 0 e 1, incluindo o tamanho zero (w ϵ {0, 1}*), porém a barra (/) representa um tal que, e o módulo de w (|w|) indica tamanho das palavras, ou seja, esta linguagem é formada por palavras com {0,1} em qualquer ordem porém somente palavras com tamanho maior que 1, resultando em L = {00, 01, 10, 11, 000, ...}

Gramáticas

As gramáticas são os mecanismos geradores das linguagens formais, e com uma gramática é possível gerar qualquer palavra de uma determinada linguagem. As gramáticas são utilizadas nos compiladores para reconhecimento de linguagens, por exemplo, se temos a linguagem L_1, sendo formada por todos os códigos em C possíveis (códigos corretos, que compilem), então a partir da gramática de L_1 deve ser possível chegar em qualquer código de um programa em C.

As gramáticas seguem as divisões proposta pela hierarquia de Chomsky, de acordo com a tabela abaixo, e estas gramáticas dividem-se em 4 tipos: **Gramática Regular, Gramática Livre de Contexto, Gramática Sensível ao Contexto e Gramática Irrestrita**.

Linguagem	**Gramática**	**Reconhecedor**
Tipo 0: Linguagem Enumerável Recursivamente	Irrestrita	Máquina de Turing
Tipo 1: Sensíveis ao Contexto	Sensíveis ao Contexto	Autômato Limitado Linearmente
Tipo 2: Livre de Contexto	Livre de Contexto	Autômato com Pilha
Tipo 3: Regulares	Regular	Autômato Finito

Uma gramática G é formada por uma quádrupla (V, T, P, S), na qual:

- **V** é um conjunto de símbolos não-terminais (ou variáveis)
- **T** é um conjunto finito de símbolos terminais disjuntos de V
- **P** é um conjunto finito de pares, denominados regras de produção tal que a primeira componente é a palavra de $(VUT)^+$ e a segunda componente é a palavra $(V \cup T)^*$
- **S** é um elemento de V, denominado símbolo inicial, ou símbolo de partida

Os símbolos de T equivalem aqueles que aparecem nos programas de uma linguagem de programação. É o alfabeto em cima do qual a linguagem é definida;

Os elementos de V são símbolos auxiliares que são criados para permitir a definição das regras da linguagem. Eles correspondem à "categorias sintáticas" da linguagem definida:

- Português: sentença, predicado, verbo, ...;
- Pascal: programa, bloco, procedimento, ...;

Uma regra de produção (α,β) é representada por $\alpha \rightarrow \beta$. Nas regras de produção definem-se as condições de geração das palavras da linguagem. O uso de uma regra de produção é chamado de derivação e a existência de uma regra de produção $\alpha \rightarrow \beta$ indica que α pode ser substituída por β sempre que α aparecer.

Na derivação de sentenças, enquanto houver símbolo não-terminal na cadeia em derivação, esta derivação não terá terminado, ou seja, a derivação acontece até gerar uma palavra.

O símbolo inicial é o símbolo através do qual deve iniciar o processo de derivação de uma sentença. Os conjuntos das variáveis (V) e dos terminais (T) devem ser disjuntos, ou seja, $V \cap T = \emptyset$, não tendo assim elementos em comum. Normalmente os elementos de T são os Terminais representados por letras minúsculas (a, b, c, d, ...). Os elementos de V são os não-terminais representados por letras maiúsculas (A, B, C, D, ...). As cadeias mistas, isto é, aquelas que contém símbolos de V e símbolos de T serão representadas por letras gregas (α, β, ...)

Quando se cria uma gramática para uma linguagem L, a partir do símbolo inicial deve ser possível "chegar" em todas as palavras pertencentes a linguagem. Isto é possível a partir das derivações realizadas a partir do símbolo inicial S. Por exemplo, para a gramática G_1 abaixo, como símbolo inicial da derivação temos o símbolo S. Para fazer uma derivação com S, devemos procurar uma regra de produção (P), na qual S está do lado esquerdo, assim só temos uma regra existente ($S \rightarrow AB$), isso significa que podemos trocar S por AB. Ainda assim temos os símbolos A e B, que não são terminais, e devem ser derivados, devendo novamente procurar regras de produção com A e B do lado esquerdo das regras, encontrando assim uma única regra para cada símbolo ($A \rightarrow a$ para A e $B \rightarrow b$, para B), resultando em uma única palavra ab.

- G1 = ({S, A, B}, {a, b}, P, S), onde:
  - P = { 1) $S \rightarrow AB$

    2) $A \rightarrow a$

    3) $B \rightarrow b$}

Mostrando a derivação de uma forma diferente podemos ter:

Símbolo inicial	Regra utilizada
S	1) $S \rightarrow AB$
AB	2) $A \rightarrow a$
a**B**	3) $B \rightarrow b$
ab	ab é uma Palavra da Linguagem

Esta derivação indica que a palavra ab faz parte da linguagem gerada pela gramática G_1. Como não temos nenhum outro caminho possível para tomar na derivação das regras, a linguagem gerada por G_1 pode ser descrita como a linguagem $L_1 = \{ab\}$

Observe agora a gramática G_2, para analisar as palavras que podem ser geradas por G_2.

- $G_2 = (\{S, A\}, \{a\}, P, S)$, onde:
  - $P = \{$ 1) $S \rightarrow aA$

    2) $A \rightarrow aA$

    3) $A \rightarrow \lambda\}$

Uma possibilidade de palavra gerada por G_2 pode ser vista usando a derivação abaixo:

Símbolo inicial	Regra utilizada
S	1) $S \rightarrow aA$
aA	3) $A \rightarrow \lambda$
a**λ**	$a\lambda = a$
a	a é uma Palavra da Linguagem

Assim, temos que a palavra a pertence à linguagem gerada por G2.

Símbolo inicial	Regra utilizada
S	1) $S \rightarrow aA$
aA	2) $A \rightarrow aA$
a**aA**	3) $A \rightarrow \lambda$
aa**λ**	$aa\lambda = a$
aa	aa é uma Palavra da Linguagem

Símbolo inicial	Regra utilizada
S	1) $S \rightarrow aA$
aA	2) $A \rightarrow aA$
a**aA**	2) $A \rightarrow aA$
aa**aA**	3) $A \rightarrow \lambda$
aaa**λ**	$aaa\lambda = a$
aaa	aaa é uma Palavra da Linguagem

Observando as derivações acima percebemos que a única possibilidade de derivar S é a regra 1, esta regra por sua vez gera um símbolo a na palavra e outro símbolo A. Este símbolo A pode ser derivado n vezes a partir da regra 2, gerando assim um novo símbolo A, mas para a derivação finalizar deve-se utilizar a regra 3 que substitui o A por um símbolo vazio (λ). Isto pode ser tratado como um loop de a, que pode gerar infinitos a's depois de gerar o primeiro a na regra 1. Assim a linguagem gerada pela gramática G_2 pode ser descrita como $L_2 = \{a^n / n > 0\}$

Para a gramática G3 abaixo, temos que partindo do símbolo inicial S geramos um único símbolo A, que também tem-se uma única possibilidade e gera os símbolos BaC, sendo que os símbolos B e C são loops para gerar quantidades de b e c respectivamente. Para B devemos ter pelo menos um b na palavra final e para C devemos ter pelo menos dois c na palavra final, resultando em uma linguagem $L_3 = \{b^m ac^{2n} \mid m,n \geq 1\}$

- G3 = ({S, A, B, C}, {a, b, c}, P, S), onde:
  - P = { 1) $S \rightarrow A$

    2) $A \rightarrow BaC$

    3) $B \rightarrow Bb$

    4) $B \rightarrow b$

    5) $C \rightarrow ccC$

    6) $C \rightarrow cc$ }

Fica como exercício mostrar as derivações para as seguintes palavras (bacc, bbacc, bbbacc, bacccc, bbacccc)

Como visto acima, as gramáticas podem ser divididas em quatro tipos, Gramática Regular, Gramática Livre de Contexto, Gramática Sensível ao Contexto e Gramática Irrestrita, sendo que o que diferencia cada uma destas gramáticas é o formato das regras de produção. Por exemplo, a regra 2 da gramática G3 ($A \rightarrow BaC$), não pode existir em uma gramática regular, podendo existir em qualquer outro tipo de gramática.

Para as Gramáticas Regulares ainda tem-se uma subdivisão em quatro tipos, também dependendo do formato das regras de produção.

Gramática Regular

Uma linguagem regular é aquela que pode ser descrito por uma gramática regular. As GR se dividem em quatro tipos: Gramática Linear à Direita (GLD), Gramática Linear à Esquerda (GLE), Gramática Linear Unitária à Direita (GLUD) e Gramática Linear Unitária à Esquerda (GLUE)

Gramática Linear à Direita

Seja G = (V, T, P, S) e sejam A e B elementos de V, e w uma cadeia de T^*, então:

- Uma Gramática é uma **GLD**, se todas as produções são da forma:
  - $A \rightarrow wB$
  - $A \rightarrow w$

Gramática Linear À Esquerda

Seja G = (V, T, P, S) e sejam A e B elementos de V, e w uma cadeia de T^*, então:

- Uma Gramática é uma **GLE**, se todas as produções são da forma:
  - $A \rightarrow Bw$
  - $A \rightarrow w$

Gramática Linear Unitária à Direita

Seja G = (V, T, P, S) e sejam A e B elementos de V, e w uma cadeia de T^*, então:

- Uma Gramática é uma **GLUD**, se todas as produções são da forma:
  - $A \rightarrow wB$
  - $A \rightarrow w$
  - Com $|w| \leq 1$

Gramática Linear Unitária à Esquerda

Seja G = (V, T, P, S) e sejam A e B elementos de V, e w uma cadeia de T^*, então:

- Uma Gramática é uma **GLUE**, se todas as produções são da forma:
  - $A \rightarrow Bw$
  - $A \rightarrow w$
  - Com $|w| \leq 1$

Dentre os quatro tipos de gramática regular o que diferencia cada um deles é a forma na qual a derivação acontece para as palavras da linguagem, sendo que as gramáticas à direita (GLD e GLUD) derivam a palavra partindo do primeiro símbolo para o último (em direção à direita), enquanto que as gramáticas à esquerda (GLE e GLUE) derivam a palavra partindo do último símbolo da palavra em direção ao primeiro (em direção à esquerda).

A diferença entre GLD e GLUD, e a diferença entre GLE e GLUE é que nas gramáticas não unitárias (GLD e GLE), uma derivação pode gerar mais de um símbolo terminal, enquanto que nas unitárias (GLUD e GLUE) a cada derivação gera-se apenas um único símbolo terminal.

Uma linguagem regular pode ser representada por qualquer um dos quatro tipos de gramáticas regulares (GLD, GLE, GLUD, e GLUE), ou seja, para uma linguagem regular deve ser possível fazer os quatro tipos de gramática regular.

Tomando a linguagem $L_1 = \{a^n bbc^m \ / \ n > 0 \text{ e } m \geq 0\}$ construiremos os quatro tipos de GR para esta linguagem.

Para a construção da GLD, devemos pensar na construção da gramática com a palavra crescendo para a direita, assim sabemos que o primeiro símbolo de uma palavra qualquer de L_1 será um a, pois n deve ser maior que zero. À direita deste primeiro símbolo a, podemos ter uma quantidade n de a's, ou seja, um loop para aumentar a quantidade de símbolos a, criando assim a regra 1 (S → aA). Ainda pensando no sentido para direita da palavra, quando não tivermos mais a na palavra, temos que a linguagem indica os dois símbolos bb obrigatoriamente em todas as palavras, seguido à direita por uma quantidade m de c, sendo que esta quantidade m pode ser zero ou infinitos c's. Como após o primeiro a, temos a obrigatoriedade de dois bb, podemos criar a regra 3 para os dois b's (A → bbC), e a regra 2 para o loop de a (A → aA). E ainda para a quantidade m de c, que pode ser vazia, criamos a regra 4 (C → cC) para o loop de c e a regra 5 (C → λ) para quando não temos nenhum c na palavra. A GLD para L1 pode ser vista abaixo.

- GLD = (V = {S, A, C}, T = {a, b, c}, P, S = S), e P possui as seguintes produções:
  - 1) S → aA
  - 2) A → aA
  - 3) A → bbC
  - 4) C → cC
  - 5) C → λ

Observa-se que todas as regras de produção estão no formato de uma GLD (A → wB ou A → w), assim podemos dizer que a gramática acima é uma Gramática Linear à Direita, porém não podemos dizer que a gramática acima é uma gramática unitária à direita (GLUD), pois a regra 3 ($A \to bbC$), não pode existir em uma GLUD, pois na GLUD deve-se ter um único símbolo em w, ou o vazio.

Para a construção da GLUD deve-se a cada regra ter um único símbolo terminal (ou um único símbolo minúsculo), seguido de uma única variável (um único símbolo maiúsculo). Assim uma Gramática Linear Unitária à Direta para $L_1 = \{a^n bbc^m / n > 0 \text{ e } m \geq 0\}$ pode ser descrita como:

- GLUD = (V = {S, A, B, C}, T = {a, b, c}, P, S = S), e P possui as seguintes produções:
    - 1) $S \to aA$
    - 2) $A \to aA$
    - 3) $A \to bB$
    - 4) $B \to bC$
    - 5) $C \to cC$
    - 6) $C \to \lambda$

Nesta GLUD, cada regra de produção possui um único símbolo terminal, seguido de um único símbolo não terminal, ou a regra com o vazio ($C \to \lambda$). Isto pode ser feito utilizando uma nova variável após o único símbolo terminal possível, por exemplo, na GLD temos a regra $A \to bbC$, que não pode existir na GLUD, assim utilizamos uma nova variável B como sendo $B \to bC$, e substituindo na regra original de $A \to bbC$ os dois últimos termos bC por B, resultando em $A \to bB$, tornando assim todas as regras de produção no formato de uma GLUD ($A \to wB$ ou $A \to w$, com $|w| \leq 1$)

Para a Gramática Linear à Esquerda, montamos as regras de produção começando do último símbolo em direção aos primeiros, uma vez que nas regras de produção o símbolo maiúsculo está do lado esquerdo do símbolo terminal ($A \to Bw$ ou $A \to w$).

A GLE para $L_1 = \{a^n bbc^m / n > 0 \text{ e } m \geq 0\}$ pode ser vista abaixo:

- GLE = (V = {S, A}, T = {a, b, c}, P, S = S), e P possui as seguintes produções:
    - 1) $S \to Sc$
    - 2) $S \to Aabb$
    - 3) $A \to Aa$
    - 4) $A \to \lambda$

Na construção da GLE deve-se observar a linguagem a partir do último símbolo possível, com direção aos primeiros, assim criamos uma regra S → Sc para representar o loop de c no final da linguagem, na qual não é necessário ter nenhum c, mas podemos ter infinitos c's. À esquerda destes c's (existentes ou não), temos dois bb, e a esquerda destes ainda um a como sendo obrigatório, assim criamos uma nova regra partindo de S com estes símbolos que são obrigatórios (S → Aabb), e a partir de A, temos duas regras, uma A → Aa para aumentar a quantidade de a no início da palavra, e A → λ para quando deseja-se terminar a derivação de A.

Para esta GLE a existência de duas regras com S (S → Sc e S → Aabb) indica que a derivação das palavras começa com S e pode tomar dois caminhos, o primeiro caminho usa a regra 1 e gera um novo S, sendo utilizada para representar a inserção de um c na palavra seguido à esquerda de um novo S. O segundo caminho usa a regra 2 e gera os dois bb's e à esquerda destes o a obrigatório, seguidos à esquerda ainda por um A.

O uso da regra 2 no início da derivação indica que não teremos c na palavra, porém à esquerda ainda temos o abb que é a menor palavra da linguagem. O uso da regra 1 indica que podemos ter um loop de c, usando quantas vezes seja necessária a regra 1, porém para finalizar a derivação devemos utilizar a regra 2 e gerar os símbolos obrigatórios da linguagem, e consequentemente gerar os possíveis a's adicionais no início.

A GLUE para $L_1 = \{a^n bbc^m \ / \ n > 0 \text{ e } m \geq 0\}$ pode ser vista abaixo:

- GLUE = (V = {S, A, B}, T = {a, b, c}, P, S = S), e P possui as seguintes produções:
    - 1) S → Sc
    - 2) S → Bb
    - 3) B → Ab
    - 4) A → Aa
    - 5) A → a

A construção de uma GLUE deve ser feita de tal forma que a cada derivação tenha-se um único símbolo não terminal e um único símbolo terminal, assim a regra 1 (S → Sc) indica que podemos iniciar a derivação de S adicionando um c no final da palavra, e mantendo o símbolo S, ou seja, a cada uso da regra 1 inserimos um c na palavra, porém é necessário ter abb em todas as palavras, sendo necessário então utilizar a regra 2 (S → Bb) para gerar o segundo b, seguido da regra 3 (B → Ab) para gerar o primeiro b de uma palavra qualquer. Após o uso da regra 3, tem-se um A na derivação e este pode ser derivado diretamente pelo único a obrigatório (regra 5, A → a), ou usar a regra 4 (A → Aa) para gerar um loop de a's, que finalizará obrigatoriamente com a regra 5.

Exemplo 2 – $L_2 = \{a^n b^{3m} / n > 0, m > 0\}$. Fazer os quatro tipos de gramática regular (GLD, GLUD, GLE, GLUE) para a linguagem L_2.

- GLD = (V = {S, A, B}, T = {a, b}, P, S = S), e P possui as seguintes produções:
    - 1) $S \rightarrow aA$
    - 2) $A \rightarrow aA$
    - 3) $A \rightarrow bbbB$
    - 4) $B \rightarrow bbbB$
    - 5) $B \rightarrow \lambda$
- GLUD = (V = {S, A, B, C, D}, T = {a, b}, P, S = S), e P possui as seguintes produções:
    - 1) $S \rightarrow aA$
    - 2) $A \rightarrow aA$
    - 3) $A \rightarrow bB$
    - 4) $B \rightarrow bC$
    - 5) $C \rightarrow bD$
    - 6) $D \rightarrow bB$
    - 7) $D \rightarrow \lambda$
- GLE = (V = {S, A, B}, T = {a, b}, P, S = S), e P possui as seguintes produções:
    - 1) $S \rightarrow Bbbb$
    - 2) $B \rightarrow Bbbb$
    - 3) $B \rightarrow Aa$
    - 4) $A \rightarrow aA$
    - 5) $A \rightarrow \lambda$
- GLUE = (V = {S, A, B, C, D}, T = {a, b}, P, S = S), e P possui as seguintes produções:
    - 1) $S \rightarrow Bb$
    - 2) $B \rightarrow Cb$
    - 3) $C \rightarrow Db$

- 4) $D \rightarrow Bb$
- 5) $D \rightarrow Aa$
- 6) $A \rightarrow Aa$
- 7) $A \rightarrow \lambda$

Exemplo 3 – Faça os quatro tipos de Gramática Regular (GLD, GLUD, GLE, GLUE) para a seguinte linguagem $L_3 = \{a^n(bc)^m a \ / \ n > 0, m > 0\}$

A GLD/GLUD/GLE/GLUE para L_3 pode ser vista abaixo:

- GLD = (V = {S, A, B}, T = {a, b, c}, P, S = S), e P possui as seguintes produções:
  - 1) $S \rightarrow aA$
  - 2) $A \rightarrow aA$
  - 3) $A \rightarrow bcB$
  - 4) $B \rightarrow bcB$
  - 5) $B \rightarrow a$
- GLUD = (V = {S, A, B, C}, T = {a, b, c}, P, S = S), e P possui as seguintes produções:
  - 1) $S \rightarrow aA$
  - 2) $A \rightarrow aA$
  - 3) $A \rightarrow bB$
  - 4) $B \rightarrow cC$
  - 5) $C \rightarrow bB$
  - 6) $C \rightarrow a$
- GLE = (V = {S, A, B}, T = {a, b, c}, P, S = S), e P possui as seguintes produções:
  - 1) $S \rightarrow Abca$
  - 2) $A \rightarrow Abc$
  - 3) $A \rightarrow Ba$
  - 4) $B \rightarrow Ba$
  - 5) $B \rightarrow \lambda$

- GLUE = (V = {S, A, B, C}, T = {a, b, c}, P, S = S), e P possui as seguintes produções:
    - 1) $S \rightarrow Aa$
    - 2) $A \rightarrow Bc$
    - 3) $B \rightarrow Cb$
    - 4) $C \rightarrow Bc$
    - 5) $C \rightarrow Ca$
    - 6) $C \rightarrow a$

Como podemos ver nos exemplos 1, 2 e 3 acima, uma Gramática Linear Unitária à Direita também é uma Gramática Linear à Direita, do mesmo jeito que uma Gramática Linear à Esquerda também é uma Gramática Linear á Esquerda, assim quando deseja-se fazer os quatro tipos de Gramática Regular pode-se descrevê-las somente utilizando os dois tipos de gramáticas unitárias (GLUD e GLUE).

Para $L_4 = \{(ab)^n(cd)^m \ / \ n > 0, m > 0\}$ podemos visualizar as duas gramáticas unitárias abaixo.

- GLUD = (V = {S, A, B, C, D}, T = {a, b, c, d}, P, S = S), e P possui as seguintes produções:
    - 1) $S \rightarrow aA$
    - 2) $A \rightarrow bB$
    - 3) $B \rightarrow aA$
    - 4) $B \rightarrow cC$
    - 5) $C \rightarrow dD$
    - 6) $D \rightarrow cC$
    - 7) $D \rightarrow \lambda$
- GLUE = (V = {S, A, B, C, D}, T = {a, b, c, d}, P, S = S), e P possui as seguintes produções:
    - 1) $S \rightarrow Ad$
    - 2) $A \rightarrow Bc$
    - 3) $B \rightarrow Ad$

- 4) B → Cb
- 5) C → Da
- 6) D → Cb
- 7) D → λ

Observando o Exemplo L_4 acima, podemos perceber que como o formato de uma regra de produção para uma GLD é A → wB, com w ϵ T* e A e B ϵ V, e o formato de uma GLUD além disto precisa ter |w| ≤ 1, uma regra que está no formato de uma GLUD também está no formato de uma GLD, o mesmo valendo para uma regra de produção que esteja no formato de uma GLUE também está no formato de uma GLE.

Outro exemplo de GLUD e GLUE para a linguagem L_5 = {w ϵ {a,b}* / a quantidade de a é impar e a quantidade de b é impar}

- GLUD = (V = {S, A, B, C}, T = {a, b}, P, S = S), e P possui as seguintes produções:
  - 1) S → aB
  - 2) S → bA
  - 3) A → aC
  - 4) A → bS
  - 4) B → bC
  - 5) B → aS
  - 6) C → aA
  - 7) C → bB
  - 8) C → λ
- GLUE = (V = {S, A, B, C}, T = {a, b}, P, S = S), e P possui as seguintes produções:
  - 1) S → Ba
  - 2) S → Ab
  - 3) A → Ca
  - 4) A → Sb
  - 4) B → Cb
  - 5) B → Sa

- 6) C → Aa
- 7) C → Bb
- 8) C → λ

Bibliografia

VIEIRA, NEWTON JOSÉ. **Introdução aos Fundamentos da Computação**. São Paulo. Pioneira Thomson Learning. 2006;

MENEZES, P. B.; DIVERIO, T. A.; **Teoria da Computação: Máquinas Universais e Computabilidade**; 3ª edição Bookman 2011

MENEZES, P. B. ; **Linguagens Formais e Autômatos**. 6ª edição. Ed. Artmed. 2011

SIPSER M. **Introdução à Teoria da Computação**. 2 ed. Cengage Learning.2007

MORET, B. M. **"Theory of Computation".** Addison-Wesley, 1998.

HOPCROFT, JOHN E.; ULLMAN, JEFFREY D.; MOTWANI, RAJEEV **Introdução à Teoria de Autômatos, Linguagens e Computação** Ed.Campus 2002

SILVA, FLAVIO SOARES CORRÊA; MELO, ANA CRISTINA VIEIRA; **Modelos Clássicos de Computação** Ed. Thomson 1ª Edição 2006