Teoria dos Autômatos e Linguagens Formais

aula 24: O lema do bombeamento

1 Introdução

- *Será que existe um autômato de pilha que reconhece a linguagem*

$$a^n\, b^n\, a^n \qquad ?$$

Bom, a resposta é não.

E hoje nós vamos ver porque.

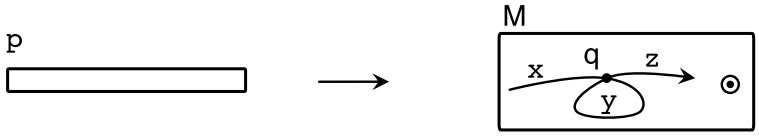
2 Lema do bombeamento

A ideia é saber se existe uma versão do lema do bombeamento para os autômatos de pilha.

Quer dizer, a linguagem $a^n b^n a^n$ é uma linguagem infinita, que possui palavras arbitrariamente grandes.

Mas, um autômato de pilha sempre tem um número finito de estados.

Logo, a computação de um autômato de pilha sobre uma palavra muito, muito grande vai acabar passando mais de uma vez por algum estado q



Sim, é verdade.

Mas, o diabo é a pilha ...

Quer dizer, o conteúdo da pilha no momento das duas passagens pelo estado q podem ser diferentes.

Daí que, nós não temos a garantia de que após a remoção (ou a duplicação) do fragmento y, a computação ainda vai terminar no estado final — (*com a pilha vazia*)

E daí, o argumento do bombeamento já não funciona mais.

Certo.

Mas a gente pode levar a pilha em consideração.

E daí, a ideia fica assim

Quer dizer,

→ *Se o autômato passar duas vezes pelo estado* q *com o mesmo conteúdo na pilha, então o fragmento* y *(lido entre as duas passagens) pode ser removido ou bombeado sem afetar o resultado final da computação.*

Sim, isso é verdade.

Mas, o diabo é que a pilha pode ficar arbitrariamente grande ...

Daí que, nós não temos a garantia de que o autômato vai estar duas vezes no mesmo estado com o mesmo conteúdo na pilha durante a computação.

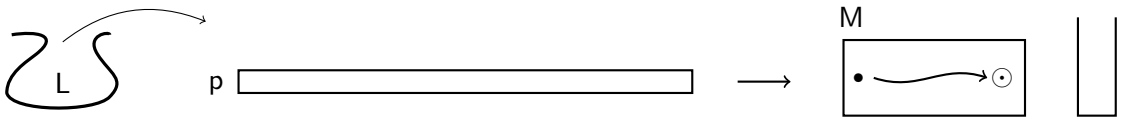
E daí, o argumento do bombeamento falha mais uma vez.

Certo.

Então a solução é pegar o diabo pelo chifre.

Quer dizer, imagine que a pilha do autômato fica arbitrariamente grande quando ele recebe as palavras da linguagem L.

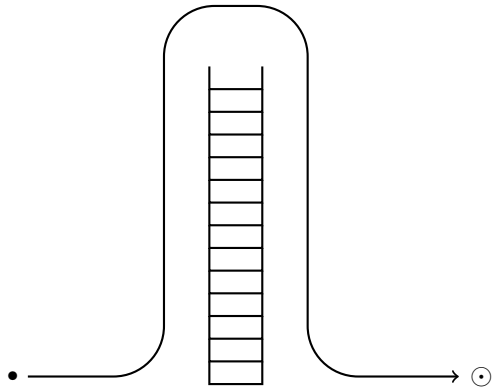
E considere uma palavra muito, muito grande da linguagem L



Legal.

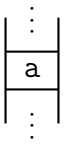
O truque é olhar para a pilha — (*ou o chifre ...*)

Quer dizer, a ideia é que a pilha vai ficar muito, muito grande durante a computação — (*e depois ela vai voltar a ficar vazia*)



E agora a gente tem que ver a repetição.

Bom, para ver a repetição a gente concentra a atenção sobre um símbolo qualquer da pilha

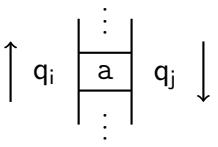


E daí, a gente observar que a computação passa duas vezes por esse ponto

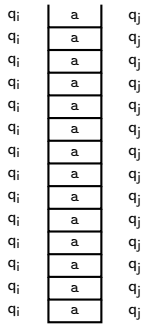
subida: o símbolo estava no topo, e o autômato colocou outro símbolo sobre ele

descida: o símbolo estava no topo, e o autômato removeu ele da pilha

E a gente anota os dois estados em que o autômato estava quando essas coisas aconteceram



Daí a gente pode imaginar que essas anotações são feitas para todos os símbolos da pilha

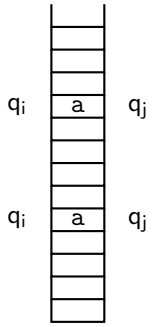


Bom, o número de estados do autômato é finito.

E o número de símbolos que podem ser colocados em uma posição da pilha é finito.

Daí que, o número de triplas (q_i, a, q_j) é finito.

Daí que, se a pilha é realmente muito, muito grande, vão haver dois pontos da pilha que tem a mesma anotação



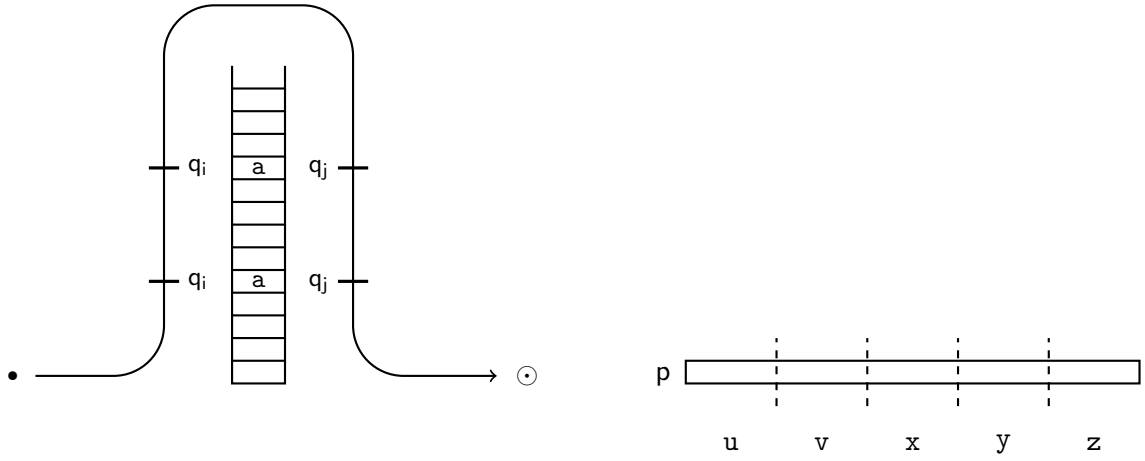
Pronto!

Agora nós vimos a repetição na computação do autômato de pilha.

E agora o resto é moleza ...

Quer dizer, o primeiro passo é quebrar a computação nos momentos de subida e descida associados a esses dois pontos.

E daí, claro, isso nos dá uma decomposição da palavra de entrada em 5 partes



Agora, é bem fácil ver que

→ *Após a remoção (ou a duplicação) dos fragmentos* `v` *e* `y` *da palavra de entrada a computação ainda termina no estado final — (com a pilha vazia)*

Quer dizer, é assim que as linguagens reconhecidas por autômatos de pilha bombeiam

$$\texttt{uvxyz} \in \mathsf{L} \quad \Rightarrow \quad \texttt{uxz},\ \texttt{uvvxyyz} \in \mathsf{L}$$

— (*onde a gente não sabe quem são* $\texttt{u}, \texttt{v}, \texttt{x}, \texttt{y}, \texttt{z}$*, e precisa considerar todas as possibilidades*)

Legal, não é?

3 Lema do bombeamento para gramáticas

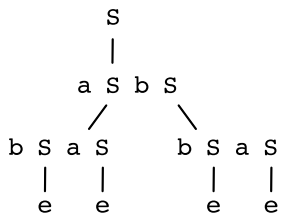
Na prática, os livros desenvolvem esse raciocínio com as gramáticas l.c., e é isso o que nós vamos ver em seguida.

Considere a seguinte gramática para a linguagem BAL

$$S \quad \to \quad \texttt{a}S\texttt{b}S \ \mid \ \texttt{b}S\texttt{a} \ \mid \ \texttt{e}$$

Como foi observado na aula passada, o fato de algumas regras possuirem duas variáveis no lado direito faz com que seja mais conveniente raciocinar sobre derivações na forma de árvore.

Por exemplo, a árvore de derivação abaixo



produz a palavra `ababba` — (*você consegue ver?*)

Agora, as coisas ficam mais interessantes quando nós observamos que uma árvore de derivação possui *subárvores* no seu interior

— Figura —

Quer dizer, qualquer variável no interior de uma árvore de derivação possui uma subárvore embaixo dela (que corresponde às regras que são aplicadas a partir daquela variável).

E nós sempre podemos trocar uma subárvore por outra, desde que as suas raízes sejam as mesmas.

Por exemplo, na passagem abaixo

— Figura —

nós trocamos a subárvore da esquerda por uma cópia da árvore original.

(*Você consegue ver?*)

E daí, nós podemos trocar a subárvore mais à esquerda outra vez por uma cópia da primeira árvore

— Figura —

Você notou que já tem uma forma de bombeamento acontecendo aqui?

Agora, só nos resta colocar as coisas no formato geral de um lema do bombeamento.

Lema do bombeamento para linguagens LC

Suponha que L é uma linguagem LC infinita.

E suponha que G é uma gramática LC que produz a linguagem L.

Ora, L é infinita mas G não é.

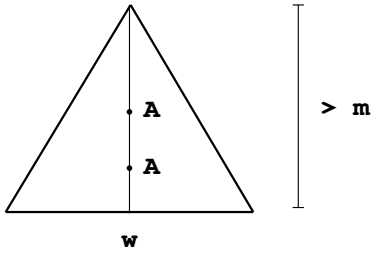
Quer dizer, G possui um número finito de variáveis (e de regras).

Isso significa que para produzir uma palavra muito grande de L, as variáveis de G vão ter que aparecer mais de uma vez durante a derivação — i.e., ao menos uma delas.

Mas nós podemos raciocinar em termos de *árvores de derivação.*

E a observação que nós podemos fazer é a seguinte

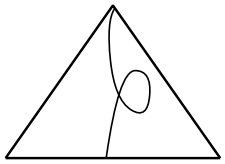
- em uma árvore de derivação muito grande, existe um caminho onde alguma variável A aparece mais de uma vez



A intuição é que isso corresponde a um laço na derivação

e que nós podemos utilzar esse laço para fazer o bombeamento na derivação.

Certo.



Mas, vamos recapitular o que nós temos até aqui.

A ideia é que

→ se p é uma palavra muito grande de L, então p deve ser produzida por uma árvore de derivação com profundidade muito grande

→ mas, uma árvore de derivação com profundidade muito grande tem um caminho que forma um laço que pode ser bombeado

— Figura —

Agora, nós precisamos entender qual é o efeito do bombeamento sobre a palavra p.

O primeiro passo é voltar para a representação da árvore com as duas ocorrências da variável A no caminho

— Figura —

Daí, nós decompomos a árvore de derivação primeiro assim

— Figura —

Quer dizer, u é a porção da palavra p que é produzida por tudo que fica à esquerda do primeiro A na árvore.

E z é a porção da palavra p que é produzida por tudo que fica à direita do primeiro A na árvore.

A parte $\mathtt{p}'$ corresponde à porção da palavra que é produzida por essa variável A.

A seguir, nós fazemos outra decomposição

— Figura —

E agora nós estamos prontos para realizar o bombeamento.

Quer dizer, a ideia é trocar a subárvore do segundo A por uma cópia da subárvore do primeiro A

— Figura —

E agora, nós podemos ver qual é o resultado do bombeamento:

- os fragmentos u e y foram duplicados

A operação inversa corresponde a trocar a subárvore do primeiro A por uma cópia da subárvore do segundo A

— Figura —

E o resultado dessa operação é

- os fragmentos u e y foram removidos

Pronto!

Agora nós estamos prontos para enunciar o lema do bombeamento para linguagens LC

⇒ Se p é uma palavra muito grande de uma linguagem LC L, então p pode ser decomposta em 5 partes

$$\mathtt{p} \quad = \quad \mathtt{u\,v\,x\,y\,z}$$

de modo que as palavras

$$p' \quad = \quad u\,x\,z \qquad\qquad \text{e} \qquad\qquad p' \quad = \quad u\,v\,v\,x\,y\,y\,z$$

também pertencem à linguagem L

De fato, como nós podemos aplicar o bombeamento mais de uma vez, qualquer palavra da forma

$$u\,v^k\,x\,y^k\,z$$

pertence à linguagem L.

4 Argumento do bombeamento

Como no caso do lema do bombeamento para linguagens regulares, o Teorema 1 é utilizado tipicamente para mostrar que certas linguagens não são LC.

A ideia é simples:

Se alguma palavra muito grande da linguagem L não pode ser bombeada como indicado no Teorema 1, então a linguagem L não é LC.

Vejamos um exemplo concreto.

Exemplo 1: Considere a linguagem $\mathtt{a}^n\mathtt{b}^n\mathtt{a}^n$, com $n \geq 0$

Intuitivamente, essa linguagem não é LC pois não há como verificar que os 3 blocos da palavra possuem o mesmo tamanho utilizando apenas uma pilha como memória.

Para demonstrar esse fato, nós assumimos que m é um número arbitrariamente grande, e consideramos a palavra

$$\mathtt{w} \quad = \quad \mathtt{a}^m\mathtt{b}^m\mathtt{a}^m$$

A seguir, é preciso considerar todas as formas como a palavra w pode ser decomposta em 5 segmentos

$$\mathtt{w} \quad = \quad \mathtt{x\,y\,z\,u\,v}$$

Após pensar um pouco, nós descobrimos que os dois casos abaixo cobrem todas as possibilidades:

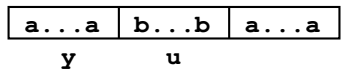
caso 1: y (ou u) possui mais de um tipo de símbolo (e.g., y = a..ab..b)

Nesse caso, é fácil ver que ao bombear os segmentos y u nós obtemos uma palavra que não tem mais o padrão a...ab...ba...a.

Logo, a palavra $w_2 = xy^2zu^2v$ não está em $a^nb^na^n$, e o bombeamento falha nesse caso.

caso 2: y e u são blocos de símbolos do mesmo tipo

Nesse caso, y e u devem ser parte dos blocos que constituem a palavra w. Por exemplo,



Logo, ao remover (ou bombear) os fragmentos y,u nós obtemos uma palavra que não possui blocos do mesmo tamanho e, portanto, não pertence à linguagem $a^nb^na^n$. Ou seja, o bombeamento também falha nesse caso.

Finalmente, como o bombeamento falha tem todos os casos, nós podemos concluir que a linguagem $a^nb^na^n$ não é LC.

◇

A seguir, nós vamos ver outros exemplos.

Exemplos:

a. 3 blocos do mesmo tamanho

Considere a linguagem

L = palavras que possuem 3 blocos de a's do mesmo tamanho

Para demonstrar que L não é uma linguagem LC, nós assumimos que m é um número arbitrariamente grande e consideramos a seguinte palavra de L

$$w = a^m\, b\, a^m\, b\, a^m$$

A seguir, é preciso considerar todas as formas como a palavra w pode ser decomposta em 5 segmentos

$$w = x\,y\,z\,u\,v$$

Como no exemplo anterior, os dois casos abaixo cobrem todas as possibilidades:

caso 1: y (ou u) contém o símbolo b

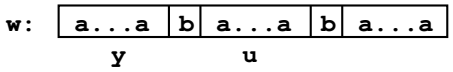
Nesse caso, a remoção dos segmentos y,u produz uma palavra que não possui 3 blocos de a's

a.....aba.....a ou a.....a

Ou seja, a palavra $w_0 = xzv$ não pertence à linguagem L, e o bombeamento falha nesse caso.

caso 2: y e u são blocos de a's

Nesse caso, y e u devem ser parte dos blocos de a's que aparecem na palavra w. Por exemplo, Logo, ao bombear (ou remover) os segmentos y,u nós obtemos uma palavra que não possui



3 blocos de a's do mesmo tamanho e, portanto, não pertence à linguagem L. Ou seja, o bombeamento também falha nesse caso.

Finalmente, como o bombeamento falha tem todos os casos, nós podemos concluir que a linguagem L não é LC.

◇

b. Considere a linguagem

CRESC = $a^i b^j a^k$, onde $i < j < k$

Para demonstrar que CRESC não é uma linguagem LC, nós assumimos que m é um número arbitrariamente grande e consideramos a palavra

$$w = a^m b^{m+1} a^{m+2}$$

A seguir, é preciso considerar todas as formas como a palavra w pode ser decomposta em 5 segmentos

$$w = xyzuv$$

Os dois casos abaixo cobrem todas as possibilidades:

caso 1: y (ou u) possui símbolos de mais de um tipo (e.g., y = a...ab...b)

Nesse caso, ao bombear os segmentos y, u nós obtemos uma palavra que não tem o padrão

$$a.....ab.....ba.....a$$

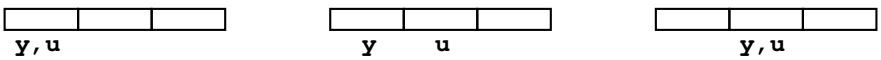
Logo, a palavra $w_2 = xy^2zu^2v$ não pertence a CRESC, e o bombeamento falha nesse caso.

caso 2: y e u são blocos de símbolos do mesmo tipo

Nesse caso, y e u estão contidos em um dos blocos que constituem a palavra w. Por exemplo,

Dessa vez, é preciso um pouco mais de cuidado, e nós vamos considerar duas situações em separado.

- Se y e u estão contidos nos dois primeiros blocos da palavra



então o bombeamento de y,u quebra o padrão crescente dos tamanhos dos 3 blocos de w (basta acrescentar um símbolo a um dos blocos e manter o bloco à sua direita inalterado para isso ocorrer)

- Se y ou u estão contidos no último bloco da palavra

então a remoção de y,u quebra o padrão crescente dos tamanhos dos 3 blocos de w (basta remover um símbolo de um dos blocos e manter o bloco à sua esquerda inalterado para isso ocorrer)

Ou seja, o bombeamento falha também no caso 2.

Finalmente, como o bombeamento falha em todos os casos, nós podemos concluir que a linguagem CRSC não é LC.

◇

c. Considere a linguagem

$\mathsf{L_p} \;=\; \mathtt{a}^{\mathtt{p}}$, onde p é um número primo

Suponha por um momento que L_p é uma linguagem livre de contexto, e considere a palavra $\mathtt{w} = \mathtt{a}^{\mathtt{p}}$, onde p é um número primo muito grande.

Então, de acordo com o Teorema 1, é possível decompor a palavra w como

$$\mathtt{w} \;=\; \mathtt{u\,v\,x\,y\,z}$$

de modo que $\mathtt{uv}^{\mathtt{k}}\mathtt{xy}^{\mathtt{k}}\mathtt{z}$ também pertence à linguagem L_p para todo $k \geq 0$.

Por outro lado, como w contém apenas o símbolo a, nós sabemos que

$$\mathtt{u} \;=\; \mathtt{a}^{\mathtt{i}} \qquad \mathtt{v} \;=\; \mathtt{a}^{\mathtt{j}} \qquad \mathtt{x} \;=\; \mathtt{a}^{\mathtt{l}} \qquad \mathtt{y} \;=\; \mathtt{a}^{\mathtt{m}} \qquad \mathtt{z} \;=\; \mathtt{a}^{\mathtt{n}}$$

onde $j + m > 0$.

Tomando $k = (i + l + n)$, segue que

$$\begin{aligned} \mathtt{uv}^{\mathtt{k}}\mathtt{xy}^{\mathtt{k}}\mathtt{z} \;&=\; \mathtt{a}^{i}\mathtt{a}^{j(i+l+n)}\mathtt{a}^{l}\mathtt{a}^{m(i+l+n)}\mathtt{a}^{n} \\ &=\; \mathtt{a}^{(i+l+n)}\mathtt{a}^{(1+j+m)} \end{aligned}$$

também pertence à linguagem L_p.

Mas isso é uma contradição, pois $(i + l + n)(1 + j + m)$ claramente é um número composto.

Logo, L_p não é uma linguagem livre de contexto.

◇

d. Considere a seguinte linguagem formada por palavras construídas com o alfabeto $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.

$\mathsf{BAL}_3 \;=\;$ palavras que contém a mesma quantidade de a's, b's e c's

Intuitivamente, não é possível verificar que os 3 tipos de símbolos aparecem na mesma quantidade utilizando uma pilha como memória. Logo, BAL_3 não deve ser LC.

Para demonstrar esse fato, nós assumimos que m é um número arbitrariamente grande, e consideramos a palavra

$$\mathtt{w} \;=\; \mathtt{a}^{\mathtt{m}}\mathtt{b}^{\mathtt{m}}\mathtt{c}^{\mathtt{m}}$$

A seguir, nós consideramos todas as formas como a palavra w pode ser decomposta em 5 segmentos

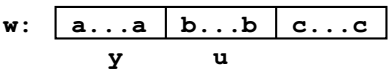
$$\mathtt{w} \;=\; \mathtt{x\,y\,z\,u\,v}$$

Existe um caso simples:

caso 1: `y` e `u` são blocos de símbolos do mesmo tipo

Nesse caso, `y` e `u` estão ambos contidos em blocos que constituem a palavra w.

Por exemplo,

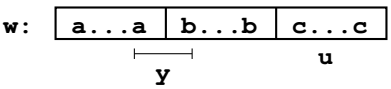


Logo, ao bombear (ou remover) os segmentos `y`,`u` a palavra deixa de ter a mesma quantidade de `a`'s, `b`'s e `c`'s, e portanto não pertence mais à linguagem BAL_3. Ou seja, o bombeamento falha nesse caso.

Mas, o próximo caso apresenta problemas.

caso 2: `y` (ou `u`) possuem mais de um tipo de símbolo

Nesse caso, os segmentos podem estar localizados como indicado no esquema abaixo



Ao bombear os segmentos `y`,`u` a palavra deixa de ter o padrão

$$\mathtt{a.....ab.....bc.....c}$$

mas isso não é relevante para a linguagem BAL_3.

E, o que é pior, caso os segmentos `y` e `u` tenham a forma

$$\mathtt{y} \;=\; \mathtt{a}^k\mathtt{b}^k \qquad\qquad \mathtt{u} \;=\; \mathtt{c}^k$$

o bombeamento ou a remoção de `y`,`u` deixa a palavra com a mesma quantidade de `a`'s, `b`'s e `c`'s. Ou seja, nós não conseguimos obter o resultado desejado dessa vez.

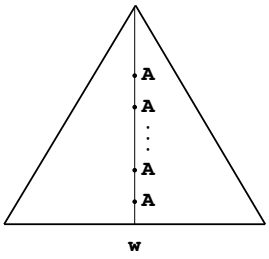
A solução consiste em utilizar uma versão um pouco mais poderosa do lema do bombeamento.

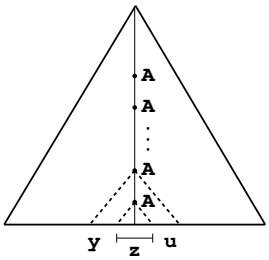
Considere uma árvore de derivação realmente muito grande, com altura bem maior do que o número de símbolos não-terminais da gramática G.

Nesse caso, os caminhos mais longos da árvore devem conter, não apenas duas, mas muitas ocorrências do mesmo símbolo A.

A observação chave é que o argumento do bombeamento apresentado na Seção **??** pode ser aplicado a qualquer par de símbolos A. Em particular, nós podemos escolher o par que fica mais embaixo no caminho.

Como indicado na figura acima, ao fazer isso os segmentos $\mathtt{y}, \mathtt{z}, \mathtt{u}$ correspondem a porções relativamente pequenas da palavra `w`. Ou seja, nós podemos assumir que $\mathtt{y}, \mathtt{u}$ não são muito grandes e não ficam muito distantes um do outro. Essa condição adicional é o que permite aplicar





o argumento bombeamento em situações como o caso 2 acima.

caso 2: (revisitado)

Utilizando a condição de que os segmentos `y` e `u` devem estar próximos, eles só podem conter parte de dois blocos da palavra `w`. Por exemplo,

Isto é, sempre existe um tipo de símbolo que não aparece nos segmentos $\mathtt{y}, \mathtt{u}$. Logo, ao bombear ou remover os segmentos $\mathtt{y}, \mathtt{u}$ a palavra deixa de ter a mesma quantidade de `a`'s, `b`'s e `c`'s, e portanto não está mais em BAL_3. Ou seja, o bombeamento falha também nesse caso.

Como o bombeamento falha em todos os casos, nós podemos concluir que a linguagem BAL_3 não é LC.

Abaixo, nós temos a versão formal do lema do bombeamento que foi utilizada nesse exemplo.

Teorema 2: (Lema do bombeamento - versão forte)

Para toda gramática livre de contexto G existem números n, m tais que toda palavra de comprimento maior do que n pode ser decomposta como

$$w \quad = \quad x\, y\, z\, u\, v$$

de modo que

i) $yu \neq \mathtt{e}$

ii) $|yzu| \leq m$

iii) a palavra $w_k = xy^k zu^k v$ pode ser gerada pela gramática G, para todo $k \geq 0$.

$\diamond$

e. Considere a linguagem

$\mathsf{L} \;=\; \mathtt{a}^m\mathtt{b}^n\mathtt{a}^m\mathtt{b}^n$, onde $m, n \geq 0$

Intuitivamente, essa linguagem não é LC pois os pares de bloco que devem ser comparados aparecem intercalados.

Para demonstrar esse fato, nós assumimos que l, k são números arbitrariamente grandes, e consideramos a palavra

$$w \;=\; \mathtt{a}^l\mathtt{b}^k\mathtt{a}^l\mathtt{b}^k$$

A seguir, é preciso considerar todas as formas como a palavra w pode ser decomposta em 5 segmentos

$$w \;=\; x\,y\,z\,u\,v$$

Os dois casos abaixo cobrem todas as possibilidades.

caso 1: y (ou u) possui símbolos de mais de um tipo (e.g., $y = \mathtt{a...ab...b}$)

Nesse caso, é fácil ver que ao bombear os segmentos y, u a palavra deixa de ter o padrão

$$\mathtt{a}.....\mathtt{ab}.....\mathtt{ba}.....\mathtt{ab}.....\mathtt{b}$$

e portanto não pertence mais à linguagem L. Ou seja, o bombeamento falha nesse caso.

caso 2: y e u são blocos do mesmo tipo de símbolo

Assumindo adicionalmente que y e u não se encontram distantes um do outro, existem basicamente duas situações a considerar

isto é, y, u fazem parte do mesmo bloco de w, ou y, u são formados por símbolos de tipos diferentes.

Em ambos os casos, a remoção ou bombeamento dos segmentos y, u fazem com que pares de blocos do mesmo tipo deixem de ter o mesmo tamanho, e portanto a palavra deixa de pertencer à linguagem L. Ou seja, o bombeamento também falha nesse caso.

Finalmente, como o bombeamento falha em todos os casos, nós podemos concluir que a linguagem L não é LC.

$\diamond$

f. Considere a linguagem

$\mathsf{DUP} \;=\;$ palavras da forma $\alpha\alpha$, onde $\alpha \in \{\mathtt{a}, \mathtt{b}\}^*$

Intuitivamente, essa linguagem não é LC pois, após a leitura da primeira cópia de α, o próximo símbolo deve ser comparado com o símbolo que se encontra na base da pilha, o que não é possível.

Para demonstrar esse fato, nós vamos escolher a palavra w com bastante cuidado. Suponha que m é um número arbitrariamente grande, e considere a palavra

$$\underbrace{\mathtt{aba}^2\mathtt{ba}^3\mathtt{b}\ldots\mathtt{ba}^m\mathtt{b}}_{\alpha}\;\underbrace{\mathtt{aba}^2\mathtt{ba}^3\mathtt{b}\ldots\mathtt{ba}^m\mathtt{b}}_{\alpha}$$

A seguir, é preciso considerar as formas como w pode ser decomposta em 5 segmentos

$$w \quad = \quad x\, y\, z\, u\, v$$

A chave aqui é observar que, como os segmentos y, u não podem estar muito distantes um do outro, eles não podem estar posicionados sobre blocos de a's do mesmo tamanho.

Logo, ao remover os segmentos y, u da palavra w, blocos de a's de um certo tamanho deixam de existir em uma das cópias de α, mas continuam a existir na outra cópia de α. (Note que isso é verdade independente do formato dos segmentos y, u.) Nesse caso, a palavra não pode mais pertencer à linguagem DUP.

Ou seja, o bombeamento falha e nós podemos concluir que DUP não é uma linguagem LC.

$\diamond$