Lema do Bombeamento para Linguagens Regulares

***Resumo*—Este trabalho é resultado de estudos referentes ao *Lema do Bombeamento para Linguagens Regulares*. Nosso objetivo é demonstrar como este método auxilia na identificação de uma determinada linguagem quanto a ser regular ou não-regular. No entanto, vale destacar que, utilizando apenas o *Lema do Bombeamento*, é possível somente determinar se uma dada linguagem é não-regular, pois para uma linguagem regular este lema representa, simplesmente, uma de suas propriedades. Em nossas demonstrações, apresentamos, a príncipio, a definição do *Lema do Bombeamento*, e, em seguida, sua prova e alguns exemplos.**

Palavras-chave - Teoria da Computação; Linguagens Regulares; Lema do Bombeamento.

I. Introdução

A Teoria da Computação abrange o estudo de modelos de computadores ou máquinas e o respectivo poder computacional destes modelos. Isto é, quais as classes de problemas que podem ser resolvidas em cada modelo e como representá-las.

O modelo de computação atual é incapaz de entender completamente a linguagem humana direta: seja falada ou escrita, dado o número enorme de possibilidades de significados e/ou acepções de uma mesma palavra, além das variações de construções de frases.

Para diminuir este distanciamento entre a língua humana (por exemplo: o português, o inglês, etc.) e a programação de computadores foram criadas as linguagens de programação. Estas são as linguagens formais, as quais procuram eliminar toda ambiguidade possível, garantindo assim que comandos e palavras reservadas tenham sempre o mesmo significado, independentemente, de onde estão localizadas no programa.

As linguagens formais são úteis em diversas áreas, mas no caso da Computação, em particular, as linguagens formais têm uma importância ímpar, pois a maioria dos profissionais da área lidam diretamente com uma, ou mais, no dia a dia. No entanto, tal importância está relacionada, de fato, às funcionalidades presentes na teoria das linguagens formais, a qual baseia-se em métodos matemáticos que propiciam a especificação e o reconhecimento de linguagens.

O Lema do Bombeamento enquadra-se nos métodos matemáticos utilizados como base pela teoria das linguagens formais. Visto que, esse lema possui a capacidade de reconhecer a classe de uma determinada linguagem. Ou seja, através do Lema do Bombeamento pode-se concluir, por exemplo, se uma dada linguagem é não-regular. Entretanto, utilizando apenas esse método, não existe a possibilidade de reconhecer se uma certa linguagem pertence a classe das linguagens regulares.

De acordo com a Hierarquia de Chomsky, as linguagens regulares constituem a classe de linguagens mais simples, sendo possível desenvolver algoritmos de reconhecimento, de geração ou de conversão, entre formalismos, de pouca complexidade, de grande eficiência e de fácil implementação. [1]

Além da utilização de autômatos finitos, existem outras maneiras para reconhecer uma linguagem regular. Ou seja, pode-se identificar se uma dada linguagem é regular antes mesmo de construir o autômato - citado anteriormente. Encontram-se diversas caracterizações adicionais que podem servir de auxílio em tais situações. Especificamente, o Lema do Bombeamento, explicita uma característica importante de toda linguagem regular. Contudo, nem toda linguagem que possui esta propriedade está presente na classe das linguagens regulares. Isto é, há linguagens com essa característica, mas que se situam nas classes das linguagens não-regulares.

Este trabalho está estruturado da seguinte forma: na Seção II apresentamos as principais características das linguagens regulares, bem como sua importância, na Seção III expomos a definição do Lema do Bombeamento, uma prova para ele e alguns exemplos, e, por fim, na Seção IV, finalizamos nossas análises, relatando as conclusões obtidas de nossos estudos.

II. Linguagens Regulares

Para uma linguagem ser considerada como linguagem regular é necessário que ela seja reconhecida por, pelo menos, um Autômato Finito Determinístico (AFD), o qual se caracteriza por sempre atingir um *estado de aceitação* - a palavra é aceita ou não. Ademais, todas as linguagens regulares podem ser expressas utilizando expressões regulares, uma forma precisa de identificar uma determinada sequência de caracteres [1].

As linguagens finitas também se caracterizam pelo fato de serem reconhecidas por um ou mais AFD's. Segundo *Newton José Vieira*, uma linguagem é dita finita se, e somente se, existe algum AFD, cujo diagrama não possui ciclos, que a indentifique. No entanto, apesar de toda linguagem finita ser uma linguagem regular, a recíproca não é verdadeira - nem toda linguagem regular é finita [2].

Vale ressaltar que as linguagens regulares representam as linguagens com o menor poder computacional na *Hierarquia de Chomsky*. Dessa forma, pode-se elaborar algoritmos, para tal linguagem, de pouca complexidade. Como apresentado na Figura 1 as linguagens regulares fazem parte do tipo 3 - a classe de linguagens mais simples - na hierarquia citada. Como o tipo três é o nível mais "baixo" ele está contido em todos os outros tipos [1].

As linguagens regulares estão presentes também numa outra "classe": linguagens formais. Tais linguagens evidenciam duas propriedades primordiais: possuem uma sintaxe e uma semântica inequívocas. De acordo com *Newton José Vieira*, as linguagens formais têm uma sintaxe bem definida, para qualquer sentença existe a possibilidade de determinar se ela

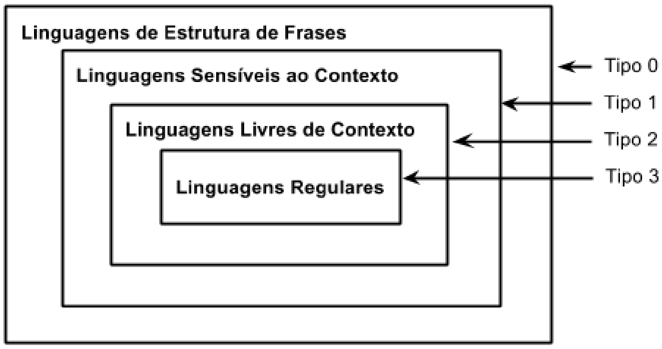

Figura 1. Hierarquia de Chomsky

pertence ou não a linguagem, e uma semântica concisa, não há sentenças sem significado ou ambíguas [2].

Apesar das inúmeras características positivas, as linguagens regulares sofrem com algumas desvantagens. A principal delas é a limitação de expressividade. Por exemplo, as fórmulas aritméticas não podem ser representadas por linguagens regulares [1].

No entanto, analisando a eficiência e a simplicidade dos algoritmos, podemos perceber que, caso haja a possibilidade de resolver um determinado problema através de uma solução regular, essa deve ser a escolha preferencial.

III. Lema do Bombeamento

O Lema do Bombeamento se enquadra como uma das caracterizações utilizadas para verificar se uma dada linguagem é regular ou não. Isto é, existem meios que em conjunto podem determinar se uma linguagem pertence à classe das linguagens regulares ou das linguagens não-regulares. Um desses meios é o próprio Lema do Bombeamento. Em suma, essas caracterizações, unidas, podem substituir os AFD's na identificação de uma linguagem regular. Ou seja, pode-se determinar se uma linguagem L pertence ou não à classe - das linguagens regulares - antes mesmo da tentativa de construir um Autômato Finito para ela.

Todas as linguagens regulares contém uma propriedade especificada pelo Lema do Bombeamento, mas essa propriedade também pode estar presente em outras linguagens - não reconhecidas por AFD's. Dessa forma, utilizando apenas este lema não é possível reconhecer a classe de uma determinada linguagem, mais especificadamente se ela é uma linguagem regular. Visto que, como citado anteriormente, existem linguagens não-regulares que possuem a propriedade individualizada pelo lema.

O teorema do Lema do Bombeamento, segundo a definição de alguns autores, como [2] [1] [3] [4], é tal que, dada uma determinada linguagem L, se L é uma linguagem regular, então existe um número k, o qual representa o comprimento do bombeamento, onde w, uma cadeia qualquer de L, tem $|w| \geq k$. Caso a afirmação anterior seja verdadeira, w pode ser dividida em três partes, $w = xyz$. Para tanto, também se faz necessário satisfazer as seguintes regras:

1. para cada $i \geq 0, xy^iz \in L$,
2. $|y| > 0$, e
3. $|xy| \leq k$.

Assim que, w é dividida em xyz, x ou z podem ser vazio, porém, de acordo com a condição 2, y não pode ser vazio. Sem a condição 2, o teorema seria verdadeiro. Note que a condição 3, diz que o tamanho de x e y seria no máximo o comprimento de k. Ela pode ser útil ao provar que que certas linguagens não são regulares [3].

Idéia da prova: (proposta por [3]) Considere um autômato finito deterministico (AFD) $M = (Q, \Sigma, \delta, i1, F)$, onde M reconhece L, e o comprimento do bombeamento representado por k, é dado pelo número de estados que compõe M. Para qualquer cadeia w em L, se esta tiver o comprimento pelo menos igual ao de k, pode ser quebrada em três partes, como indicado anteriormente, em xyz. Sendo assim as três condições serão satisfeitas. A primeira análise que pode ser realizada para a prova, é verificar todas as cadeias de L, se elas não possuírem no mínimo o tamanho de k, o teorema se torna verdadeiro por *vacuidade*, ou seja, a maior cadeia da linguagem L é menor que o k, e as três condições se verificam para cadeias com comprimento no mínimo k [3]. A próxima etapa se dá pela situação de w em L ter o comprimento igual ou maior que k. Considerando a sequência de estados pelos quais M atravessa quando está computando a entrada w, supomos que a partir do estado inicial do autômato finito, percorre-se vários estados, repetindo duas vezes um determinado estado, até que atinja o final de w. Sabendo-se que w está contida em L, conclui-se a aceitação de w por M [3].Se dissermos que n possui comprimento de w, e representa os estado que foram ativados durante o consumo de w, teremos como comprimento $n + 1$, porque ouve uma repetição em um determinado estado. Sabemos que n é no mínimo p, logo $n + 1$ vai ser maior que a quantidade de estados do autômato em questão. Este é um exemplo conhecido como o princípio da casa dos pombos, que acontece quando se tem k pombos para se colocar em $k-1$ casas. Desta forma, alguma casa irá receber mais que um pombo [3]. Por fim, para comprovar o teorema vamos dividir a cadeia w nas três partes, que são x, y e z, e observaremos o autômato fictício da Figura 2. Os caracteres de x aparecem antes do estado $e8$, pois o fragmento pertencente a y aparece nas n ocorrências de $e8$, e o de z vem logo após $e8$, levando até o estado final, $e12$ [3].

Como vimos, três situações devem ser consideradas para afirmarmos que uma dada linguagem pode ser bombeada. Vamos ver o porquê de w satisfazer as três condições do teorema. Analisando a Figura 2, vemos que x leva do estado inicial até o $e8$, o y leva para o mesmo $e8$ e z leva para o estado de aceitação. Assim conclui-se que M aceita a cadeia xy^iz, para qualquer $i \geq 0$. Se tomamos um i=3, a entrada para M seria $xyyyz$, e a cadeia, neste formato, será aceita por M. Continuando com a analogia, se tomarmos um $i = 0$,

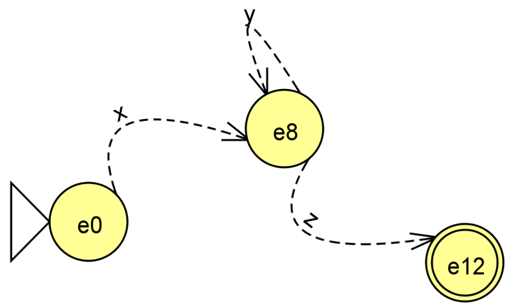

Figura 2. Como as cadeias x, y e z afetam M - Adaptado de [3]

teremos $xy^0z = xz$, sabendo que y^0 será igual a λ, a qual representa a palavra vazia. Desta forma, a cadeia formada por xz continua sendo aceita pelo autômato M, assim fica estabelecida a condição um. Tomando a condição dois, teremos que $|y| > 0$, logo y era parte de w que aparecia entre as duas ocorrências diferente do estado $e8$. Para termos a condição três como válida, é necessário fazer com que $e8$ seja a primeira repetição da sequência. Considerando o princípio da casa dos pombos, os $p + 1$ estados devem ocorrer por meio de uma repetição, para que $|x + y| \leq k$ [3].

A. *Exemplo para uma Linguagem Regular*

Sabemos que a linguagem $L = \{0^n1|n \geq 0\}$ é regular, portanto vamos aplicar o lema do bombeamento sobre ela para verificar se esta linguagem pode ser bombeada.

Primeiramente devemos demonstrar que existe uma constante k, tal que para toda palavra $w \in L$, temos $|w| \geq k$. Desta maneira podemos subdividir a cadeia w da seguinte forma $w = xyz$.

Verificando a propriedade da linguagem L percebemos que todas as cadeias pertencentes a L contém uma quantidade nula ou infinita de 0s e a cadeia deve obrigatoriamente terminar com um único simbolo 1. Então, w deve seguir essa regra. Exemplo: L = {1, 01, 0000001, ...}

Decompondo w em xyz, conforme as regras do lema, $|y| > 0$ e $|xy| \leq k$, iremos supor $w = 0^k1$. Assim sendo, temos:

$$x = \lambda,\ y = 0^k \text{ e } z = 1;$$

Após a decomposição de w podemos realizar o bombeamento na variável i da subdivisão xy^iz. Definindo $k = 2$ teremos a seguinte cadeia $w = 001$, a qual, de acordo com a decomposição, acarreta em $x = \lambda$, $y = 00$ e $z = 1$. Daí:

- Para $i = 0$, teremos: $xy^0z = 1$. A cadeia pertence a linguagem L;
- Para $i = 1$, teremos: $xy^1z = 001$. A cadeia continua pertencendo a linguagem L;
- Para $= 2$, teremos: $xy^2z = 00001$. Esta cadeia também pertence a L.

Dessa forma, podemos observar que a linguagem pode ser bombeada e mesmo assim as cadeias geradas continuam pertencendo a esta linguagem. Porém nós utilizamos uma linguagem que já era conhecida como regular. Visto que, o lema do bombeamento não pode provar essa situação, ele apenas é um dos passos para tal prova - como explicitado no ínicio da seção.

B. *Exemplo para uma Linguagem Não-Regular*

Com a finalidade de apresentarmos, na prática, o Lema do Bombeamento, iremos demonstrar como identificar uma linguagem não-regular.

Dada a linguagem L, tal que, $L = \{0^n1^n2^n|n \geq 0\}$, devemos demonstrar que existe uma constante k, a qual para toda palavra $w \in L$, temos $w \geq k$. Visto que, desta maneira, tornamos possível a representação de w da seguinta forma: $w = xyz$. Sendo assim, podemos prosseguir para a verificação se a linguagem possuí ou não uma propriedade determinada pelo Lema do Bombeamento:

- Primeiramente, verificamos qual a propriedade da linguagem em questão. Logo, temos que, todas as cadeias pertencentes a L, contém a mesma quantidade de 0s, 1s e 2s. Portanto, w, deve seguir essa regra. Exemplo: L = {012, 001122, 000111222, ...};
- Em seguida devemos decompor w em xyz. Para tanto, devemos respeitar as normas estabelecidas pela definição do lema. Ou seja, $|y| > 0$ e $|xy| \leq k$. E iremos supor $w = 0^k1^k2^k$:
  - y não pode ser igual a 1^k, porque, dessa forma, $|xy| = k + k \geq k$;
  - y também não pode ser igual a 2^k, pelo mesmo motivo da situação anterior: $|xy| = k + k + k \geq k$;
- Portanto, existe apenas uma solução: y representar uma parcela de 0^k. Isto é, $|y|$ pode ser igual a k, a $k - 1$, ou até mesmo, igual a 1.

Daí, se y representará uma parcela de 0^k, então x tanto pode ser λ quanto uma determinada parcela de 0^k, dependendo do tamanho do y. Logo, podemos admitir uma constante t, tal que $t > 0$, e $y = 0^t$. Assim, temos $x = 0^{k-t}$. Portanto, a decomposição final é:

$$w = 0^{k-t}0^{ti}1^k2^k$$

Com apenas um teste comprovamos que a cadeia w - decomposta - não pertence a L. Ou seja, existem várias maneiras de provar que a linguagem não possui a propriedade do Lema do Bombeamento, porém basta um bombeamento para termos essa prova. Suponha $i = 0$. Teremos a seguinte cadeia de caracteres:

$$w = 0^{k-t}0^{t0}1^k2^k$$
$$w = 0^{k-t}1^k2^k$$

Logo a quantidade de 0s já não é mais a mesma que a de 1s e 2s. Comprovando, assim, que w não pertence a linguagem L, e, principalmente, que L é uma linguagem não-regular.

IV. Conclusão

Os estudos realizados, com a finalidade de elaborar este trabalho, mostraram que existem outros caminhos para a classificação das linguagens, quanto a serem regulares ou não-regulares, sem a necessidade de se utilizar Autômatos Finitos.

Neste artigo analisamos o impacto do Lema do Bombeamento sobre as linguagens regulares e podemos notar que seu objetivo é muito interessante, pois visa “tentar” determinar se uma dada linguagem é regular ou não. No entanto, nossas conclusões levam a crer que este lema estaria mais próximo de ser um passo intermediário para a identificação das linguagens do que de uma etapa final, na qual encontramos o resultado esperado. De maneira alguma estamos menosprezando este método, visto que sua solução é uma parte muito importante no conjunto de características utilizados na identificação citada acima.

REFERÊNCIAS

[1] P. B. Menezes, *Linguagens Formais e Autômatos*. Editora Sagra-Luzzatto, 1998.

[2] N. J. Vieira, *Introdução aos fundamentos da computação: linguagens e máquinas*. Thomson Learning, 2006.

[3] M. Sipser, *Introdução à Teoria da Computação: Tradução da 2ª edição norte-americana (trad. Ruy José Guerra Barreto de Queiroz)*. São Paulo: Thomson Learning, 2007.

[4] H. R. Lewis and C. H. Papadimitriou, *Elementos de Teoria da Computação (trad. Edson Furmankiewicz)*, 2nd ed. Porto Alegre: Bookman, 2000.