Teoria da Computação **FCT-UNL, 2º semestre 2023-2024**

Notas 8: Linguagens não regulares e o lema da bombagem

Autor: João Ribeiro

Introdução

Nestas notas discutimos a existência de linguagens não regulares – linguagens que estão além do poder dos AFDs e AFNs. O lema da bombagem providencia uma maneira de demonstrarmos formalmente que uma dada linguagem não é regular.

8.1 Será que existem linguagens não regulares?

Até agora temo-nos focado em desenhar AFDs e AFNs para várias linguagens, e também em estudar as propriedades gerais das linguagens regulares (as linguagens reconhecidas por AFDs/AFNs). É natural questionarmos se existem linguagens além do alcance destes modelos de computação. Vamos usar as noções de conjunto contável e não contável para dar uma resposta inicial a esta pergunta.

Teorema 8.1 *Seja Σ um conjunto finito não vazio. Então, existem linguagens não regulares sobre Σ.*

Demonstração: Não é difícil ver que o conjunto de todos os AFDs com alfabeto Σ é contável. Como cada linguagem regular tem um AFD que a reconhece, isto implica que o conjunto das linguagens regulares também é contável. Por outro lado, o conjunto de *todas* as linguagens sobre Σ é $\mathcal{P}(\Sigma^*)$, que é não contável pelo Teorema de Cantor. Concluímos que existem linguagens não regulares sobre Σ. ■

O Teorema 8.1 mostra que, de certa forma, o conjunto das linguagens regulares é um subconjunto minúsculo de todas as linguagens! Por outro lado, a solução que este providencia não é totalmente satisfatória. Dada uma linguagem L que acreditamos ser não regular, seria muito útil ter acesso a técnicas que permitam demonstrar isto formalmente.

Por exemplo, consideremos a linguagem

$$L = \{0^n 1^n \mid n \in \mathbb{N}\} = \{\varepsilon, 01, 0011, 000111, \dots\}.$$

É instrutivo tentar construir um AFD ou AFN que reconheça L. Ao reflectirmos, deparamo-nos com a seguinte situação: Intuitivamente, o AFD que queremos construir tem de contar o número de 0s que já viu, de forma a comparar com o número de 1s que vêm a seguir. Então, o estado actual da computação (que é a "memória" do AFD) tem de codificar informação sobre o número de 0s já visto. Isto levanta problemas, pois o AFD tem um número fixo de estados, mas o número de 0s

antes do 1 pode ser arbitrariamente grande. Parece, então, muito provável que L não seja regular! Mas este argumento baseado na necessidade do AFD "contar" é apenas informal (não justificámos, por exemplo, porque é necessário "o estado actual da computação... codificar informação sobre o número de 0s já visto..."). Por exemplo, a linguagem

$$L' = \{w \in \{0,1\}^* \mid w \text{ contém o mesmo número de ocorrências de 01 e 10}\}$$

também parece, intuitivamente, precisar de um AFD que conte ocorrências de substrings, mas é possível mostrar que L' é regular!

Como podemos, então, demonstrar formalmente que uma linguagem não é regular?

8.2 Padrões de linguagens regulares: O lema da bombagem

O lema da bombagem captura formalmente alguns padrões satisfeitos por linguagens regulares. Como veremos adiante, estes padrões são fáceis de manipular. Em muitos casos, é fácil mostrar que uma dada linguagem não satisfaz estes padrões, o que nos leva a concluir que a linguagem não é regular.

Seja, então, L uma linguagem regular. Por definição, existe um AFD M que reconhece L. Vamos tentar inferir alguns padrões satisfeitos pelas palavras suficientemente longas aceites por M. Seja k o número de estados de M. Vamos considerar uma qualquer string $w \in L$ de comprimento $n \geq k$. Seja $(q_0, q_1, \ldots, q_k, \ldots, q_n)$ a sequência de estados de M gerada por w. Como $w \in L$, sabemos que q_n é um estado final de M. Uma observação simples, mas muito importante, é a seguinte: Como a sequência $(q_0, q_1, \ldots, q_k)$ dos primeiros k estados percorridos consiste em $k+1 > k$ estados, segue que existe pelo menos um estado de M que aparece repetido nesta sequência.[1] Chamemos de q o estado repetido, e sejam i e j índices da sequência acima tais que $q_i = q_j = q$ com $0 \leq i < j \leq k$.

A observação acima permite-nos dividr w em três partes consecutivas com propriedades interessantes. A primeira parte, $x = w_1 w_2 \ldots w_i$, corresponde à sequência de estados $(q_0, q_1, \ldots, q_i = q)$. A segunda parte, $y = w_{i+1} \ldots w_j$, corresponde à sequência de estados $(q_i = q, q_{i+1}, \ldots, q_j = q)$, *que começa e acaba no mesmo estado* q. Finalmente, a terceira parte, $z = w_{j+1} \ldots w_n$, corresponde à sequência de estados $(q_j = q, q_{j+1}, \ldots, q_n)$. Note-se que podemos ter $x = \varepsilon$ ou $z = \varepsilon$, mas temos sempre $y \neq \varepsilon$. Esta divisão é ilustrada na Figura 8.1.

Coleccionamos agora algumas propriedades das partes x, y, e z de w. Primeiro, como $i < j \leq k$, temos sempre $|xy| \leq k$, onde relembramos que k é o número de estados de M. Segundo, como mencionado acima, temos sempre $y \neq \varepsilon$, ou, por outras palavras, $|y| > 0$, pois $i < j$. Finalmente, temos a seguinte propriedade de "bombagem": Se removermos a parte y, resultando na string $w' = xz$, temos que $w' \in L$. Isto acontece pois a computação de x termina em $q_i = q$ e a computação de z a partir de q termina em q_n, que é um estado final de M. Esta propriedade verifica-se também se duplicarmos y, resultando na string $xy^2z = xyyz$, pois tanto a computação do primeiro y como do segundo começa e acaba em q, e portanto a computação de z terminará em q_n, que é final.

[1] Esta é uma aplicação do princípio do pombal: Se existem p pombos e $p-1$ poleiros, então existe um poleiro com pelo menos dois pombos. Apesar deste princípio ser extremamente simples e óbvio, é também muito poderoso. Mais exemplos interessantes da aplicação do princípio do pombal podem ser encontrados nesta página da Wikipedia.

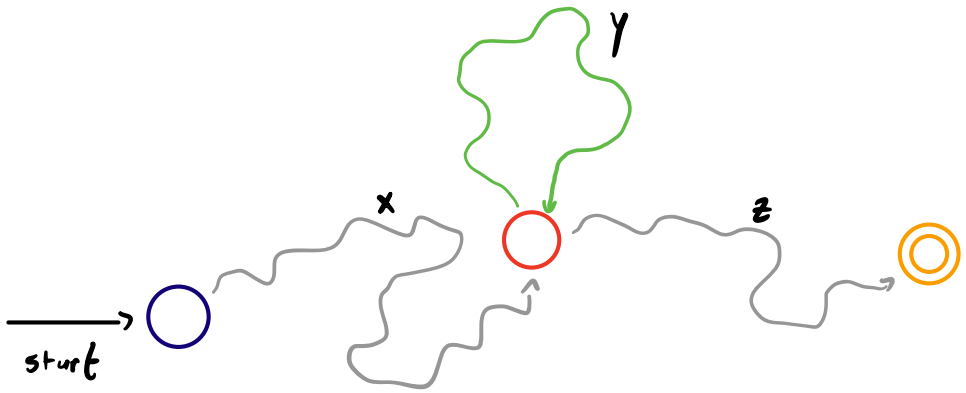

Figure 8.1: Ilustração da sequência de computação do AFD M com input $w = xyz$. O círculo azul representa o estado inicial q_0 de M e o círculo laranja representa o estado final q_n. A sequência de estados associada a y começa e acaba no mesmo estado q de M, representado a vermelho.

Podemos generalizar esta propriedade ainda mais e concluir que, se introduzirmos i cópias de y (isto é, *bombeamos* y), para qualquer $i \in \mathbb{N}$, então a string xy^iz também é aceite por M, e então $xy^iz \in L$.

As conclusões desta discussão são resumidas no seguinte lema.

Lema 8.1 (Lema da bombagem ou "pumping lemma") *Seja L uma linguagem regular. Então, existe $p \in \mathbb{N}$ (o comprimento de bombagem ou "pumping length") tal que qualquer $w \in L$ de tamanho $|w| \geq p$ pode ser escrito como $w = xyz$ com x, y, e z satisfazendo as seguintes condições:*

1. *$|xy| \leq p$;*
2. *$|y| > 0$;*
3. *$xy^iz \in L$ para qualquer $i \in \mathbb{N}$.*

O lema da bombagem para linguagens regulares foi demonstrado pela primeira vez no trabalho seminal de Rabin e Scott [RS59] com que já nos deparámos, e independentemente também por Bar-Hillel, Perles, e Shamir [BPS61] como caso especial de um teorema mais geral.

8.2.1 Exemplos de aplicação do lema da bombagem

Estes exemplos são adaptados de [Sip13, Section 1.4].

Exemplo 8.1 Consideramos mais uma vez a linguagem

$$L = \{0^n1^n \mid n \in \mathbb{N}\}.$$

Vamos demonstrar que L não é regular usando o lema da bombagem.

Suponhamos, com vista a uma contradição, que L é regular. Seja, então, $p \in \mathbb{N}$ garantido pelo lema da bombagem. Consideramos a palavra $w = 0^p1^p$. Como $w \in L$ e $|w| \geq p$, o lema da bombagem garante que podemos escrever $w = xyz$ com x, y, e z satisfazendo as seguintes condições:

1. $|xy| \leq p$;
2. $|y| > 0$;
3. $xy^iz \in L$ para qualquer $i \in \mathbb{N}$.

Como w começa com p 0s, a condição (1) implica que xy é uma substring de 0^p. Dado isto, a condição (2) implica que $y = 0^j$ para algum $j > 0$. Consideremos, então, a string $w' = xyyz = 0^{p+j}1^p$. Por um lado, a condição (3) com $i = 2$ garante que $w' \in L$. Mas, por outro lado, temos $p + j \neq p$ pois $j > 0$, e então $w' \notin L$. Como chegámos a uma contradição, concluímos que L não pode ser regular.

Exemplo 8.2 Seja L a linguagem

$$L = \{w \in \{0,1\}^* \mid w \text{ contém o mesmo número de 0s e de 1s}\}.$$

Vamos demonstrar que L não é regular usando o lema da bombagem.

Suponhamos, com vista a uma contradição, que L é regular. Seja, então, $p \in \mathbb{N}$ garantido pelo lema da bombagem. Consideramos a palavra $w = 0^p1^p$. Como $w \in L$ (pois tem p 0s e p 1s) e $|w| \geq p$, o lema da bombagem garante que podemos escrever $w = xyz$ com x, y, e z satisfazendo as seguintes condições:

1. $|xy| \leq p$;
2. $|y| > 0$;
3. $xy^iz \in L$ para qualquer $i \in \mathbb{N}$.

Como w começa com p 0s, a condição (1) implica que xy é uma substring de 0^p. Dado isto, a condição (2) implica que $y = 0^j$ para algum $j > 0$. Consideremos, então, a string $w' = xyyz = 0^{p+j}1^p$. Por um lado, a condição (3) com $i = 2$ garante que $w' \in L$. Mas, por outro lado, temos $p + j \neq p$ pois $j > 0$. Portanto, w' não tem o mesmo número de 0s e de 1s e então $w' \notin L$. Como chegámos a uma contradição, concluímos que L não pode ser regular.

Exemplo 8.3 Seja L a linguagem

$$L = \{0^n1^m \mid n \in \mathbb{N} \wedge m \in \mathbb{N} \wedge n > m\}.$$

Vamos demonstrar que L não é regular usando o lema da bombagem, desta vez bombeando 0 vezes.

Suponhamos, com vista a uma contradição, que L é regular. Seja, então, $p \in \mathbb{N}$ garantido pelo lema da bombagem. Consideramos a palavra $w = 0^{p+1}1^p$. Como $w \in L$ e $|w| \geq p$, o lema da bombagem garante que podemos escrever $w = xyz$ com x, y, e z satisfazendo as seguintes condições:

1. $|xy| \leq p$;
2. $|y| > 0$;
3. $xy^i z \in L$ para qualquer $i \in \mathbb{N}$.

Como w começa com $p+1$ 0s, a condição (1) implica que xy é uma substring de 0^p. Dado isto, a condição (2) implica que $y = 0^j$ para algum $j > 0$. Consideremos, então, a string

$$w' = xy^0 z = xz = 0^{p+1-j}1^p.$$

Por um lado, a condição (3) com $i = 0$ garante que $w' \in L$. Mas, por outro lado, temos $p+1-j \leq p$ pois $j > 0$, e então $w' \notin L$. Como chegámos a uma contradição, concluímos que L não pode ser regular.

Exemplo 8.4 Seja L a linguagem

$$L = \{0^{n^2} \mid n \in \mathbb{N}\}.$$

Vamos demonstrar que L não é regular usando o lema da bombagem.

Suponhamos, com vista a uma contradição, que L é regular. Seja, então, $p \in \mathbb{N}$ garantido pelo lema da bombagem. Consideramos a palavra $w = 0^{p^2}$. Como $w \in L$ e $|w| \geq p$, o lema da bombagem garante que podemos escrever $w = xyz$ com x, y, e z satisfazendo as seguintes condições:

1. $|xy| \leq p$;
2. $|y| > 0$;
3. $xy^i z \in L$ para qualquer $i \in \mathbb{N}$.

A condição (2) implica que $y = 0^j$ para algum $j > 0$. A condição (1) implica também que $j \leq p$. Consideremos, então, a string $w' = xy^2 z = 0^{p^2+j}$. Por um lado, a condição (3) com $i = 2$ garante que $w' \in L$. Mas, por outro lado, temos

$$p^2 + j \leq p^2 + p < p^2 + 2p + 1 = (p+1)^2,$$

pois $j \leq p$, e

$$p^2 + j > p^2,$$

pois $j > 0$. Estas duas inequações garantem que $w' \neq 0^{k^2}$ para qualquer $k \in \mathbb{N}$, e portanto $w' \notin L$. Como chegámos a uma contradição, concluímos que L não pode ser regular.

8.2.2 Atenção!

Uma linguagem que satisfaz as condições do lema da bombagem (Lema 8.1) não é necessariamente regular! Um exemplo detalhado de uma linguagem não regular que satisfaz as condições do lema da bombagem encontra-se nesta página da Wikipedia.

Existe outra caracterização necessária e suficiente de linguagens regulares (o teorema de Myhill-Nerode), cuja demonstração é deixada como desafio nesta cadeira.

8.3 Para explorar

Recomendamos a leitura de [LP97, Section 2.4] e [Sip13, Section 1.4].

Discussões sobre Pumping Lemma vs Myhill-Nerode: 1 e 2.

References

[BPS61] Yehoshua Bar-Hillel, Micha Perles, and Eli Shamir. On formal properties of simple phrase structure grammars. *Zeitschrift fur Phonetik, Sprachwissenschaft, und Kommunikationsforschung*, 14:143–172, 1961.

[LP97] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall PTR, USA, 2nd edition, 1997.

[RS59] Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[Sip13] Michael Sipser. *Introduction to the Theory of Computation*. CEngage Learning, 3rd edition, 2013.