

Teoria da Computação

Aula Teórica 17:
Linguagens não regulares.

António Ravara

Departamento de Informática, Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa

13 de Maio de 2019

Como determinar se uma linguagem é regular?

- ▶ Como saber se dada linguagem, apresentada sem se recorrer a uma expressão regular (*i.e.*, informalmente – em língua natural – ou como um conjunto), é ou não regular?
- ▶ Se não se conseguir definir directamente uma expressão regular, pode-se usar as propriedades das linguagens regulares:
São fechadas para as operações de concatenação, união, intersecção, diferença (entre duas), fecho de Kleene (entre outras).

Considerem-se os naturais em representação decimal

É regular a linguagem $\{n \in \mathbb{N}_0 \mid n \% 2 = 0 \vee n \% 3 = 0\}$?

É regular a linguagem $\{n \in \mathbb{N}_0 \mid n \% 2 = 0 \vee n \% 3 = 0\}$?

Seja $\Sigma \stackrel{\text{def}}{=} \{0, 1, \dots, 9\}$. Um natural é uma sequência de algarismos.

- ▶ Considerando que se omitem sempre zeros à esquerda, a seguinte expressão regular define a linguagem dos naturais:

$$N \stackrel{\text{def}}{=} 0 + \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \Sigma^*$$

- ▶ Os divisíveis por 2 (os pares) são as sequências de algarismos que terminam num algarismo par:

$$P \stackrel{\text{def}}{=} \Sigma^* \{0, 2, 4, 6, 8\}$$

- ▶ Para retirar os zeros à esquerda, podemos definir a linguagem

$$\mathcal{L}_1 \stackrel{\text{def}}{=} \mathcal{L}(N) \cap \mathcal{L}(P)$$

que é regular porque é a intersecção de linguagens regulares.

- ▶ Como um natural é divisível por 3 se e só se a soma dos seus algarismo é divisível por 3, define-se um AFD que aceita sequencias de algarismos cuja soma é divisível por 3.

Seja $\mathcal{L}_2$ a linguagem desse AFD.

A linguagem é $\mathcal{L}_1 \cup \mathcal{L}_2$, que é regular por $\mathcal{L}_1$ e $\mathcal{L}_2$ o serem.

Todas as linguagens são regulares?

- ▶ Uma linguagem é regular se for denotada por uma expressão regular.
- ▶ Uma linguagem é regular se for obtida por operações de fecho sobre linguagens regulares
- ▶ Pelo Teorema de Kleene, se houver um AFD que reconhece uma dada linguagem, então ela é regular.

Teorema de Kleene

Uma linguagem é regular se, e só se, existe um AFD que a reconhece.

Todas as linguagens são regulares?

- ▶ Como saber se dada linguagem é regular, quando não se conseguem aplicar os métodos referidos?
- ▶ O contra-recíproco do Teorema de Kleene é também útil:

Uma linguagem não é regular se, e só se, não existe um AFD que a reconhece

- ▶ Para mostrar que dada linguagem é regular basta encontrar um autómato finito (AFN ou AFD) que a reconheça.
- ▶ Mas para mostrar que dada linguagem NÃO é regular tem que se provar que NÃO EXISTE NENHUM autómato finito (AFN ou AFD) que a reconheça.

Quando não se consegue definir um autómato finito (AFN ou AFD) que reconheça a linguagem, como ter a certeza se existe ou não um autómato que a reconhece?

As seguintes linguagens são regulares?

- ▶ Conjunto das expressões aritméticas com o mesmo número de parêntesis esquerdos e direitos.
- ▶ Conjunto das palavras sobre o alfabeto $\{a, b\}$ que:
 - ▶ têm duas vezes mais *as* que *bs*;
 - ▶ têm um ou mais *as* seguidos do mesmo número de *bs*.
 $\{ab, aabb, aaabbb, \dots\}$

Para verificar se são regulares tem que se determinar se existe ou não um autómato finito que a reconheça.

Como proceder?

Quando não se consegue fazer o autómato e/ou se desconfia que não existe, tenta-se fazer uma PROVA POR ABSURDO.

A linguagem $\{a^k b^k \mid k \in \mathbb{N}\}$ é regular?

Procura-se construir o AFD (chame-se-lhe D):

- ▶ Seja S um conjunto de estados tal que $\#S = n \in \mathbb{N}_0$ (tem que se fixar um n).
- ▶ Para que D reconheça a linguagem dada, a palavra $a^n b^n$ tem que ser aceite por D .
- ▶ Tem que existir um “caminho” (uma sequência) de estados que permite “executar” a palavra (a partir do estado inicial e até a um estado final).
- ▶ Assume-se que $s_1 s_2 \dots s_{2n+1}$, sendo s_1 o estado inicial e s_{2n+1} um estado final, é um caminho associado à execução de $a^n b^n$.
- ▶ Logo, $s_1 s_2 \dots s_{n+1}$ é o caminho para executa a^n .
- ▶ Como D tem apenas n estados, na sequência $s_1 s_2 \dots s_{n+1}$ há (pelo menos) um estado que ocorre duas vezes.
Considere-se então $1 \leq i < j \leq n+1$ tal que $s_i = s_j$;
O caminho para executar a^n é então $s_1 \dots s_i \dots s_j \dots s_{n+1}$.

A linguagem $\{a^k b^k \mid k \in \mathbb{N}\}$ é regular?

Assume-se que $s_1 s_2 \dots s_{2n+1}$ é o caminho para executar $a^n b^n$, sendo que existem i e j com $1 \leq i < j \leq n+1$ tal que $s_i = s_j$.

- ▶ Para executar a palavra a^{n+j-i} basta então tomar o caminho $s_1 \dots s_i \dots s_j s_{i+1} \dots s_j \dots s_{n+1}$.
- ▶ Então $s_1 \dots s_i \dots s_j s_{i+1} \dots s_j \dots s_{n+1} s_{n+2} \dots s_{2n+1}$ é o caminho para executar $a^{n+j-i} b^n$, que não pertence à linguagem porque tem um número de a s diferente do de b s!
- ▶ Como s_1 o estado inicial e s_{2n+1} um estado final, a palavra $a^{n+j-i} b^n$ é aceite pelo autómato.

Mostrou-se que qualquer AFD que aceita palavras da forma $a^k b^k$ com $k \in \mathbb{N}$, também aceita palavras da forma $a^k b^l$ com $k \in \mathbb{N} \wedge l \in \mathbb{N} \wedge k \neq l$.

Conclusão

Como é impossível construir um AFD que aceite exactamente a linguagem $\{a^k b^k \mid k \in \mathbb{N}\}$, esta não é regular.

Lema da bombagem para linguagens regulares

Intuição

Notem-se as seguintes propriedades das linguagens regulares:

- ▶ Se a linguagem é infinita, o AFD que a reconhece tem ciclos e a expressão regular que a denota tem o operador estrela.
- ▶ Para determinar se dada palavra pertence a uma linguagem usa-se uma quantidade limitada e fixa à partida de memória, que depende na linguagem, não da palavra.

Enuncia-se rigorosamente a ideia com o Lema da bombagem para linguagens regulares.

Lema da bombagem para linguagens regulares

Se a linguagem $\mathcal{L}$ é regular, então existe $n \in \mathbb{N}$ tal que qualquer palavra $w \in \mathcal{L}$ que tenha pelo menos n símbolos pode ser re-escrita como $w = xyz$ com:

1. $y \neq \epsilon$;
2. xy tem no máximo n símbolos;
3. $xy^iz \in \mathcal{L}$, para cada $i \geq 0$.

Esboço da prova

Como $\mathcal{L}$ é regular existe um AFD D que a reconhece. Considere-se que D tem n estados e tome-se uma palavra $w \in \mathcal{L}(D)$ que tenha pelo menos n símbolos ($w = w_1 \dots w_n$). O caminho para aceitar w tem pelo menos $n + 1$ estados, logo pelo menos um estado repete-se: existem i e j com $1 \leq i < j \leq n + 1$ tal que $s_i = s_j$.

Tome-se $y = w_i \dots w_j$ (palavra que “leva” de s_i a s_j , que são o mesmo estado): (i) $y \neq \epsilon$ porque $i < j$; (ii) xy tem no máximo n símbolos, porque $j \leq n + 1$; (iii) y pode ser repetida zero ou mais vezes e $xy^iz \in \mathcal{L}(D)$, pois volta-se sempre ao mesmo estado.

Aplicação do Lema da bombagem para linguagens regulares

Pode-se usar o lema para fazer um jogo entre dois participantes: um tenta mostrar que dada linguagem é regular e o outro que a linguagem não o é.

1. Um dos jogadores fixa $\mathcal{L}$ e n ;
2. o oponente escolhe $w \in \mathcal{L}$ com pelo menos $n + 1$ símbolos;
3. o primeiro propõe x, y, z tal que $w = xyz$;
4. se o segundo encontra i tal que $xy^iz \notin \mathcal{L}$, ganha; senão perde.

Exemplo: $\mathcal{L} \stackrel{\text{def}}{=} \{a^k b^k \mid k \in \mathbb{N}\}$ não é regular

Assumem-se dois jogadores P_1 e P_2 ; o primeiro tenta mostrar que a linguagem é regular.

1. P_1 fixa $n > 0$ e P_2 escolhe $a^n b^n$ com $2n$ símbolos.
2. P_1 define $x = a^k, y = a^j, z = b^n$ com $k + j = n$ e $j > 0$.
3. P_2 mostra que se a linguagem é regular então $xy^2z \in \mathcal{L}$, que é uma palavra com mais as que bs.

Note-se que mesmo que P_1 escolha $k + j < n$, P_1 perde...

Aplicação do Lema da bombagem para linguagens regulares

Exemplo: $\mathcal{L} \stackrel{\text{def}}{=} \{a^k \mid k \text{ primo}\}$ não é regular

1. P_1 fixa $n > 0$ e P_2 escolhe a^m com $m > n$ símbolos.
2. P_1 define $x = a^p, y = a^q, z = a^r$, com $p, r \geq 0, q > 0$ e $m = p + q + r$ primo.
3. P_2 mostra que se a linguagem é regular então $xy^n z \in \mathcal{L}$; mas $p + nq + r$ tem que ser primo, para qualquer n ; escolhendo $n = p + 2q + r + 2 \geq 4$, obtém-se

$$\begin{aligned} p + nq + r &= p + (p + 2q + r + 2)q + r \\ &= p + pq + 2qq + rq + 2q + r \\ &= p + pq + 2qq + 2q + rq + r \\ &= (q + 1)(p + 2q + r) \end{aligned}$$

logo, $p + nq + r$ foi factorizado em dois naturais, $q + 1 > 1$ e

$1 < p + 2q + r < n$ (pois $q > 0$), pelo que não pode ser primo!