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

Notas 4: Linguagens regulares

Autor: João Ribeiro

Introdução

Nestas notas estudamos a classe das linguagens regulares – as linguagens aceites por AFDs. Estudamos como a noção de regularidade se comporta face a várias operações sobre conjuntos.

4.1 Linguagens regulares

Começamos por definir formalmente o que é uma linguagem regular.

Definição 4.1 (Linguagem regular) *Seja Σ um conjunto finito. Uma linguagem $L \subseteq \Sigma^*$ diz-se* regular *se existe um AFD M tal que $L = L(M)$.*

De outra forma, dizemos que uma linguagem L é regular se esta é aceite por algum AFD. O nosso estudo de AFDs focar-se-á em tentar entender:

- Que linguagens *são* regulares (e, nesse caso, desenhar AFDs que as aceitem);
- Que linguagens *não são* regulares, e demonstrá-lo de forma rigorosa.

4.2 Algumas propriedades de linguagens regulares

Começamos por (tentar) estabelecer algumas propriedades das linguagens regulares. Por exemplo, se L_1 e L_2 são duas linguagens regulares, será que a união $L_1 \cup L_2$ também é regular? E como é que as linguagens regulares se comportam relativamente a outras operações sobre conjuntos?

Teorema 4.1 *Se L é regular, então o seu complemento $\overline{L}$ também é regular.*

Demonstração: Começamos por descrever informalmente a ideia subjacente. Como L é regular, existe um AFD $M = (S, \Sigma, \delta, s, F)$ com função de transição δ total tal que M aceita L. Gostaríamos de criar, a partir de M, um outro AFD M' que aceita $\overline{L}$. Como sabemos que uma string $w \in L$ se e só se a sua computação em M termina num estado de aceitação, a nossa ideia é criar M' transformando os estados finais de M em estados não finais, e vice-versa.

Mais cuidadosamente, seja $M' = (S, \Sigma, \delta, s, F')$ o AFD onde $F' = S \setminus F$. Vamos mostrar que M' aceita $\overline{L}$, isto é, que $L(M') = \overline{L}$. Para verificarmos isto, temos de mostrar que $w \in L$ se e só se $w \notin L(M')$. Consideramos os dois sentidos desta equivalência separadamente:

- Se $w \in L$, então a computação de M no input w termina num estado de F por definição de M. As sequências de estados gerada por w em M e M' são iguais, portanto a computação de M' no input w também termina num estado de F. Como os estados de F não são estados finais de M', segue que $w \notin L(M')$.
- Se $w \notin L$, então a computação de M no input w termina num estado de $S \setminus F$ por definição de M. Analogamente ao caso anterior, temos que a computação de M' no input w também termina num estado de $S \setminus F$. Como estes são precisamente os estados finais de M', concluímos que $w \in L(M)$.

Em suma, mostrámos que $w \in L$ se e só se $w \notin L(M')$, e portanto $L(M') = \overline{L}$. Concluímos que $\overline{L}$ é regular. ■

Teorema 4.2 *Se L_1 e L_2 são regulares, então $L_1 \cup L_2$ também é regular.*

Demonstração: Como L_1 e L_2 são regulares, existem AFDs $M_1 = (S_1, \Sigma, \delta_1, s_1, F_1)$ e $M_2 = (S_2, \Sigma, \delta_2, s_2, F_2)$ com funções de transição totais que reconhecem L_1 e L_2, respectivamente. Queremos construir um AFD M a partir de M_1 e M_2 que reconhece $L_1 \cup L_2$. Intuitivamente, a estratégia que gostaríamos de seguir seria, dado um input w, simular as computações de M_1 e M_2 em w lado-a-lado, e aceitar w exactamente quando pelo menos um de M_1 e M_2 aceita w.

Para implementarmos a estratégia acima através do novo AFD $M = (S, \Sigma, \delta, s, F)$, precisamos de a cada momento saber os estados actuais de M_1 e M_2 na computação. Isto sugere que os estados de M deveriam ser *pares* de estados de M_1 e M_2, cada par representando o estado actual dos dois AFDs. Por outras palavras, queremos escolher $S = S_1 \times S_2$. Se M se encontrar no estado $(q_1, q_2) \in S_1 \times S_2$ num dado ponto da computação e ler o símbolo $c \in \Sigma$, como devemos actualizar o estado de M? Como queremos simular a computação de M_1 e M_2, deveríamos actualizar q_1 para $\delta_1(q_1, c)$ e q_2 para $\delta_2(q_2, c)$. Definimos então a função de transição δ como

$$\delta((q_1, q_2), c) = (\delta_1(q_1, c), \delta_2(q_2, c)).$$

Para o estado inicial s de M, escolhemos o par de estados iniciais de M_1 e M_2, isto é, $s = (s_1, s_2)$. Resta-nos especificar o conjunto de estados finais de M. Queremos aceitar os inputs cujas computações terminem num estado final de M_1 ou num estado final de M_2. Segue que devemos escolher F como o conjunto de pares (q_1, q_2) tal que $q_1 \in F_1$ ou $q_2 \in F_2$, ou seja,

$$F = (F_1 \times S_2) \cup (S_1 \times F_2).$$

Agora que especificámos M, argumentamos que $L(M) = L_1 \cup L_2$. Seja $w \in \Sigma^*$ qualquer. Denotamos a sequência de estados gerada por w em M_i por $(r_0^{(i)}, r_1^{(i)}, \ldots, r_n^{(i)})$ para $i \in \{1, 2\}$. Se $w \in L_1 \cup L_2$, pela definição de M_1 e M_2 temos que $r_n^{(1)} \in F_1$ ou $r_n^{(2)} \in F_2$. Relembrando a definição de F,

concluímos então que $\delta(w) = (r_n^{(1)}, r_n^{(2)}) \in F$. Como w é arbitrário, segue que $L_1 \cup L_2 \subseteq L(M)$. Por outro lado, se $w \notin L_1 \cup L_2$, então temos $r_n^{(1)} \notin F_1$ e $r_n^{(2)} \notin F_2$, o que significa que $(r_n^{(1)}, r_n^{(2)}) \notin (F_1 \times S_2) \cup (S_1 \times F_2) = F$, e portanto $w \notin L(M)$. Logo, $L(M) \subseteq L_1 \cup L_2$, e concluímos que $L(M) = L_1 \cup L_2$. ■

Teorema 4.3 *Se L_1 e L_2 são regulares, então $L_1 \cap L_2$ também é regular.*

Demonstração: Exercício para o leitor. ■

4.3 Obstáculos...

Ao tentarmos demonstrar mais algumas propriedades de linguagens regulares, deparamo-nos com alguns obstáculos.

Conjectura 4.1 *Se L_1 e L_2 são regulares, então $L_1 \circ L_2$ também é regular.*

Tentativa de demonstração: Como L_1 e L_2 são regulares, existem AFDs M_1 e M_2 que reconhecem L_1 e L_2, respectivamente. Temos de construir um AFD M que reconhece a concatenação $L = L_1 \circ L_2$. Após alguma reflexão, uma estratégia natural seria "concatenar" os AFDs M_1 e M_2, interpretando os estados finais de M_1 como o estado inicial de M_2. No entanto, isto levanta problemas, pois poderemos estar a definir várias transições diferentes para o mesmo símbolo a partir do mesmo estado, algo que não é permitido na definição de AFDs... Será que conseguimos ultrapassar este obstáculo?

Conjectura 4.2 *Se L é regular, então L^* também é regular.*

Tentativa de demonstração: Como L é regular, existe um AFD M que reconhece L. Temos de construir um AFD M' que reconhece o fecho de Kleene L^*, que, intuitivamente, consiste em sequências finitas de concatenações de palavras em L. Uma estratégia natural seria "ligar" os estados finais de M ao seu estado inicial. No entanto, a implementação mais directa desta estratégia esbarra no facto de termos, potencialmente, de ter várias transições diferentes para o mesmo símbolo a partir do mesmo estado...

4.4 Para explorar

Aconselhamos a leitura de [Sip13, Section 1.1] e [LP97, Section 2.1].

References

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

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