ROR(L)

- DFA con stesso stato iniziale
- D' = D
  - Legge input
  - (Hanno stesso alfabeto)
  - Avanza di tanti simboli quanti la posizione attuale rispetto alla fine
  - Scambia i valori
  - Avanza al successivo input
- Questo per tutti i simboli

Data la tupla di un DFA (quintupla):

$$(Q, \Sigma, \delta, q_0, F)$$

con Q = stati, Σ = alfabeto, δ= funz. di transizione, q_0 = stato iniziale, F = insieme di stati finali

Funzione di transizione:

$\delta(q, ab) = \delta(s, ba)$

Lo stato finale viene raggiunto quando hai scambiato tutti i simboli

Estesa:

- stesso stato iniziale → $r_0 = q_0$
- $\delta(r_0, a) = r_n$
- $\delta(r_i, w_i) = r_n \dots r_1$
- $r_i \in F$

Continuando:

- stesso stato iniziale → $r_1 = q_1$
- $\delta(r_1, a) = r_{n-1}$
- $\delta(r_{i+1}, w_{i+1}) = r_{n-1} \dots r_2$
- $r_{i+1} \in F$

Classica funzione di transizione DFA:

$$\delta(stato, input) = (stato\ che\ raggiungi, output)$$

2. Sia A un linguaggio, e sia $DROPOUT(A)$ come il linguaggio contenente tutte le stringhe che possono essere ottenute togliendo un simbolo da una stringa di A:

$$DROPOUT(A) = \{xz \mid xyz \in A \text{ dove } x, z \in \Sigma^* \text{ e } y \in \Sigma\}.$$

Mostrare che la classe dei linguaggi regolari è chiusa rispetto all'operazione $DROPOUT$, cioè che se A è un linguaggio regolare allora $DROPOUT(A)$ è un linguaggio regolare.

DROPOUT = linguaggio regolare che toglie un simbolo ogni volta dalla stringa.

- Stesso stato iniziale
- Funzione di transizione
  - Aggiunge uno stato "pozzo" oppure uno stato "scorciatoia" che ad ogni input "salta" un simbolo

- $\delta(q, ab) = (q, b)$ quando attraversa lo stato corretto
- Stesso stato finale

Siccome per chiusura dei linguaggi regolari il tuo automa può esistere, il linguaggio è regolare.

Riferimento logico: https://brainly.com/question/38483265

1. (12 punti) Data una parola $w \in \Sigma^*$, definiamo $evens(w)$ come la sottosequenza di w che contiene solo i simboli in posizione pari (iniziando a contare da 1). Per esempio, $evens(INDICEPARI) = NIEAI$. Dimostra che se $L \subseteq \Sigma^*$ è un linguaggio regolare allora anche il linguaggio

$$evens(L) = \{evens(w) \mid w \in L\}$$

è un linguaggio regolare.

Soluzione: Se L è un linguaggio regolare, allora sappiamo che esiste un DFA $A = (Q, \Sigma, \delta, q_0, F)$ che riconosce L. Costruiamo un ε-NFA A' che accetta il linguaggio $evens(L)$, aggiungendo un flag $0, 1$ agli stati di A, dove flag 0 corrisponde a "simbolo in posizione pari" e flag 1 corrisponde a "simbolo in posizione dispari". La funzione di transizione è fatta in modo da alternare i flag 0 e 1 dopo ogni transizione dell'automa. Se lo stato corrente ha flag 0, l'automa consuma il prossimo simbolo della stringa di input e prosegue nello stesso stato che raggiungerebbe A dopo aver consumato lo stesso simbolo. Se lo stato corrente ha flag 1, l'automa prosegue con una ε-transizione verso uno degli stati raggiungibili dall'automa A consumando un simbolo: simulando in questo modo il fatto che i simboli in posizione dispari vanno "saltati".

Formalmente, $A' = (Q', \Sigma, \delta', q_0', F')$ è definito come segue.

- $Q' = Q \times \{0, 1\}$.
- L'alfabeto Σ rimane lo stesso.
- $\delta'((q, 0), a) = \{(\delta(q, a), 1)\}$. Con il flag 0 l'automa consuma un simbolo di input ed ha una sola alternativa possibile: lo stato $\delta(q, a)$ raggiunto da A dopo aver consumato a. Il flag diventa 1.
- $\delta'((q, 1), \varepsilon) = \{(q', 0) \mid \text{esiste } a \in \Sigma \text{ tale che } \delta(q, a) = q'\}$. Con il flag 1 l'automa procede nondeterministicamente con una ε-transizione verso uno degli stati raggiungibili da A dopo aver consumato un simbolo arbitrario.
- $q_0' = (q_0, 1)$. Lo stato iniziale corrisponde allo stato iniziale di A con flag 1 (simbolo in posizione dispari).
- $F' = F \times \{0, 1\}$. L'insieme degli stati finali di A' corrisponde all'insieme degli stati finali di A con qualsiasi flag.

Per dimostrare che A' riconosce il linguaggio $evens(L)$, data una parola $w = w_1 w_2 \dots w_n$, dobbiamo considerare due casi.

- Se $w \in L$, allora esiste una computazione di A che accetta la parola. Di conseguenza, esiste una computazione di A che accetta la parola:

  $$s_0 \xrightarrow{w_1} s_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} s_n$$

  con $s_0 = q_0$ e $s_n \in F$. Se w è di lunghezza pari, la computazione

  $$(s_0, 1) \xrightarrow{\varepsilon} (s_1, 0) \xrightarrow{w_2} (s_2, 1) \xrightarrow{\varepsilon} (s_3, 0) \xrightarrow{w_4} (s_4, 1) \xrightarrow{\varepsilon} \dots \xrightarrow{w_n} (s_n, 1)$$

  è una computazione accettante per A' sulla parola $evens(w) = w_2 w_4 w_6 \dots w_n$, perché $(s_n, 1) \in F'$. Se w è di lunghezza dispari, allora la computazione accettante per A' su $evens(w)$ è

  $$(s_0, 1) \xrightarrow{\varepsilon} (s_1, 0) \xrightarrow{w_2} (s_2, 1) \xrightarrow{\varepsilon} (s_3, 0) \xrightarrow{w_4} (s_4, 1) \xrightarrow{\varepsilon} \dots \xrightarrow{w_{n-1}} (s_{n-1}, 1) \xrightarrow{\varepsilon} (s_n, 1).$$

- Se w è accettata dal nuovo automa A', allora esiste una computazione accettante che ha la forma

  $$(s_0, 1) \xrightarrow{\varepsilon} (s_1, 0) \xrightarrow{w_1} (s_2, 1) \xrightarrow{\varepsilon} (s_3, 0) \xrightarrow{w_2} (s_4, 1) \xrightarrow{\varepsilon} \dots \xrightarrow{w_n} (s_n, 1),$$

  se $(s_n, 1)$ è uno stato finale, oppure la forma

  $$(s_0, 1) \xrightarrow{\varepsilon} (s_1, 0) \xrightarrow{w_1} (s_2, 1) \xrightarrow{\varepsilon} (s_3, 0) \xrightarrow{w_2} (s_4, 1) \xrightarrow{\varepsilon} \dots \xrightarrow{w_n} (s_n, 1) \xrightarrow{\varepsilon} (s_{n+1}, 0),$$

quando è necessaria una ε-transizione finale per raggiungere uno stato finale. In entrambi i casi possiamo costruire una computazione accettante per A sostituendo le ϵ-transizioni con transizioni che consumano un carattere. Nel primo caso si ottiene una computazione che ha la forma

$$s_0 \xrightarrow{u_1} s_1 \xrightarrow{w_1} s_2 \xrightarrow{u_2} s_3 \xrightarrow{w_2} s_4 \xrightarrow{\varepsilon} \dots \xrightarrow{w_n} s_n,$$

e accetta una parola $u = u_1 w_1 u_2 w_2 \dots u_n w_n$. Nel secondo caso si ottiene una computazione

$$s_0 \xrightarrow{u_1} s_1 \xrightarrow{w_1} s_2 \xrightarrow{u_2} s_3 \xrightarrow{w_2} s_4 \xrightarrow{\varepsilon} \dots \xrightarrow{w_n} s_n \xrightarrow{u_{n+1}} s_{n+1},$$

che accetta la parola $u = u_1 w_1 u_2 w_2 \dots u_n w_n u_{n+1}$. In entrambi i casi $evens(u) = w$, come richiesto.

Mostra che L è regolare: https://cs.stackexchange.com/questions/1331/how-to-prove-a-language-is-regular

1. Considera la seguente funzione da $\{0,1\}^*$ a $\{0,1\}^*$:

$$\text{stutter}(w) = \begin{cases} \varepsilon & \text{se } w = \varepsilon \\ aa.\text{stutter}(x) & \text{se } w = ax \text{ per qualche simbolo } a \text{ e parola } x \end{cases}$$

Dimostra che se L è un linguaggio regolare sull'alfabeto $\{0,1\}$, allora anche il seguente linguaggio è regolare:

$$\text{stutter}(L) = \{\text{stutter}(w) \mid w \in L\}.$$

Se con simboli:

- $Q = Q_1 \; X \; Q_2$
- $q_0 = q_0{}'$
- $\delta\big((q,a),\sigma\big) = (\delta(q,aa),\sigma)$
  - dove σ (sigma) è la transizione che stai compiendo coi simboli
  - (il fatto di aggiungere "a" ad "aa", o formalizzato o spiegato a parole)
  - da i=1, vale per i=2, i=3, … i=n
- $\Sigma = \Sigma'$
- $\mathrm{F} = \mathrm{F}_1 \; X \; F_2$

Se con parole:

- Come sopra, ma spieghi dicendo a parole tutto
- Balbetti aggiungendo una "a" ad ogni "a" già presente

Esempio TM:

1. *Una* macchina di Turing bidimensionale *utilizza una griglia bidimensionale infinita di celle come nastro. Ad ogni transizione, la testina può spostarsi dalla cella corrente ad una qualsiasi delle quattro celle adiacenti. La funzione di transizione di tale macchina ha la forma*

$$\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{\uparrow, \downarrow, \rightarrow, \leftarrow\},$$

dove le frecce indicano in quale direzione si muove la testina dopo aver scritto il simbolo sulla cella corrente.

Dimostra che ogni macchina di Turing bidimensionale può essere simulata da una macchina di Turing deterministica a nastro singolo.

Normalmente, seguo questa logica:

- Parto dall'inizio con una TM a nastro singolo
  - Devo simulare con la TM a nastro singolo la variante
  - $TM_{nastro\ singolo} = TM_{variante}$

Il verso è dalla TM a nastro singolo alla TM variante

- Parto dalla funzione di transizione
- Svolgo tutti gli stati, sapendo che la TM a nastro singolo si muove a ← e a → (a sx e a dx)

1. Sostituisce w con la configurazione iniziale $\#\#w\#\#$ e marca con $\hat{}$ il primo simbolo di w.
2. Scorre il nastro finché non trova la cella marcata con $\hat{}$.
3. Aggiorna il nastro in accordo con la funzione di transizione di B:
   - Se $\delta(r,a) = (s,b,\rightarrow)$, scrive b non marcato sulla cella corrente, sposta $\hat{}$ sulla cella immediatamente a destra. Poi sposta di una cella a destra tutte le marcature con un pallino. Se in qualsiasi momento S sposta una marcatura sopra un $\#$, S scrive un blank marcato al posto del $\#$ e sposta il contenuto del nastro da questa cella fino al $\#\#$ finale, di una cella più a destra.
   - Se $\delta(r,a) = (s,b,\leftarrow)$, scrive b non marcato sulla cella corrente, sposta $\hat{}$ sulla cella immediatamente a sinistra. Poi sposta di una cella a sinistra tutte le marcature con un pallino. Se in qualsiasi momento S sposta una marcatura sopra un $\#$, S scrive un blank marcato al posto del $\#$ e sposta il contenuto del nastro da questa cella fino al $\#\#$ iniziale, di una cella più a sinistra.
   - Se $\delta(r,a) = (s,b,\uparrow)$, scrive b marcato con un pallino nella cella corrente, e sposta $\hat{}$ sulla prima cella marcata con un pallino posta a sinistra della cella corrente. Se questa cella marcata non esiste, aggiunge una nuova riga composta solo da blank all'inizio della configurazione.
   - Se $\delta(r,a) = (s,b,\downarrow)$, scrive b marcato con un pallino nella cella corrente, e sposta $\hat{}$ sulla prima cella marcata con un pallino posta a destra della cella corrente. Se questa cella non esiste, aggiunge una nuova riga composta solo da blank alla fine della configurazione.

Struttura:

$\delta(leggi, input) = (stato, output, simbolo)$

Quando mi sposto devo considerare di non andare oltre ai limiti (cancelletti/inizio/fine) e posso usare nuovi simboli (uno qualsiasi) per "far capire" alla TM (che non ha RAM o altro) dove si trovava e andare così avanti, descrivendo a parole quello che vogliamo farle fare.

Si conclude dicendo:

- se accetta, allora accetta (stato di accettazione raggiunto)
- altrimenti rifiuta

2. (12 punti) Dati due DFA, considera il problema di determinare se esiste una stringa accettata da entrambi.

(a) Formula questo problema come un linguaggio $AGREE_{\text{DFA}}$.

(b) Dimostra che $AGREE_{\text{DFA}}$ è decidibile.

Decidibilità → Usa un qualcosa di decidibile per descrivere L → normalmente, una TM

L'esercizio dice: hai due DFA, entrambi devono accettare la stringa.

Quindi → uso due TM M_1, M_2 (normalmente, quando scrivo il linguaggio, mi basta dire

M: *dove M è una TM* …

1) $AGREE_{DFA} = \{\langle M_1, M_2\rangle, M_1, M_2$ accettano entrambe il linguaggio → $L(M1) = L(M2)\}$

2) Usiamo la descrizione di un oggetto decidibile per dimostrare che il mio linguaggio è decidibile → lo esprimo con una TM

M_1, M_2 *sono due* TM, w *input*

- M_1 legge w
- Scrive simboli partendo dallo stato iniziale simulando il DFA D_1
- Arriva allo stato finale terminando l'input
- Se M accetta, allora accetta
- Altrimenti rifiuta

∪ = l'unione è un'operazione regolare (si dice che è chiusa)

Simile per M_2

- M_2 legge w
- Scrive simboli partendo dallo stato iniziale simulando il DFA D_2
- Arriva allo stato finale terminando l'input
- Se M accetta, allora accetta
- Altrimenti rifiuta

Se entrambi sono arrivati alla fine, entrambi i DFA sottostanti hanno accettato la stringa

(Si chiama = dare una descrizione a livello implementativo / ad alto livello di ciò che fa la macchina)

Essendo che L è decidibile, allora avendo usato una TM che termina (DFA termina sempre), allora il linguaggio è decidibile.