Esercizi di Automi e Linguaggi Formali

Soluzioni PDA - Linguaggi Context-Free

Gabriel Rovesti
Università degli Studi di Padova

Anno Accademico 2024-2025

Esercizio 1

Costruisci un PDA per il linguaggio $\{a^i b^j c^k \mid i = j$ ***oppure*** $j = k\}$

Soluzione

Questo linguaggio rappresenta stringhe composte da sequenze di 'a' seguite da sequenze di 'b' seguite da sequenze di 'c', dove il numero di 'a' deve essere uguale al numero di 'b' oppure il numero di 'b' deve essere uguale al numero di 'c'.

Per riconoscere questo linguaggio, dobbiamo costruire un PDA che verifichi una delle due condizioni (i = j oppure j = k). Possiamo farlo usando l'unione di due PDA, ciascuno che verifica una delle condizioni.

Approccio

Costruiremo un PDA non deterministico che all'inizio sceglie quale condizione verificare:

1. Nella prima scelta, verifica che $i = j$ (contando le 'a' e le 'b')
2. Nella seconda scelta, verifica che $j = k$ (contando le 'b' e le 'c')

Definizione formale del PDA

Il PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ dove:

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$ è l'insieme degli stati
- $\Sigma = \{a, b, c\}$ è l'alfabeto di input
- $\Gamma = \{a, b, \$\}$ è l'alfabeto dello stack
- q_0 è lo stato iniziale
- $F = \{q_4, q_8\}$ è l'insieme degli stati finali
- δ è la funzione di transizione definita come segue:

Funzione di transizione δ

Inizializzazione e scelta del percorso:

- $\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$), (q_5, \$)\}$ (scelta non deterministica)

Percorso 1 (verifica $i = j$):

- $\delta(q_1, a, X) = \{(q_1, aX)\}$ per ogni $X \in \Gamma$ (memorizza tutte le 'a')
- $\delta(q_1, b, a) = \{(q_2, \varepsilon)\}$ (inizia a verificare $i = j$ consumando 'b' per ogni 'a')
- $\delta(q_2, b, a) = \{(q_2, \varepsilon)\}$ (continua a consumare 'b' per ogni 'a')
- $\delta(q_2, b, \$) = \{(q_3, \$)\}$ (passa a q_3 quando finiscono le 'a' ma ci sono ancora 'b')
- $\delta(q_2, c, \$) = \{(q_3, \$)\}$ (passa a q_3 quando iniziano le 'c')
- $\delta(q_3, b, \$) = \{(q_3, \$)\}$ (consuma tutte le 'b' restanti)
- $\delta(q_3, c, \$) = \{(q_3, \$)\}$ (consuma tutte le 'c')
- $\delta(q_3, \varepsilon, \$) = \{(q_4, \$)\}$ (stato finale se $i = j$)

Percorso 2 (verifica $j = k$):

- $\delta(q_5, a, X) = \{(q_5, X)\}$ per ogni $X \in \Gamma$ (ignora 'a')
- $\delta(q_5, b, X) = \{(q_5, bX)\}$ per ogni $X \in \Gamma$ (memorizza tutte le 'b')
- $\delta(q_5, c, b) = \{(q_6, \varepsilon)\}$ (inizia a verificare $j = k$ consumando 'c' per ogni 'b')
- $\delta(q_6, c, b) = \{(q_6, \varepsilon)\}$ (continua a consumare 'c' per ogni 'b')
- $\delta(q_6, \varepsilon, \$) = \{(q_7, \$)\}$ (trasizione quando finiscono le 'b')
- $\delta(q_7, c, \$) = \{(q_7, \$)\}$ (consuma le 'c' extra se presenti)
- $\delta(q_7, \varepsilon, \$) = \{(q_8, \$)\}$ (stato finale se $j = k$)

Rappresentazione grafica del PDA

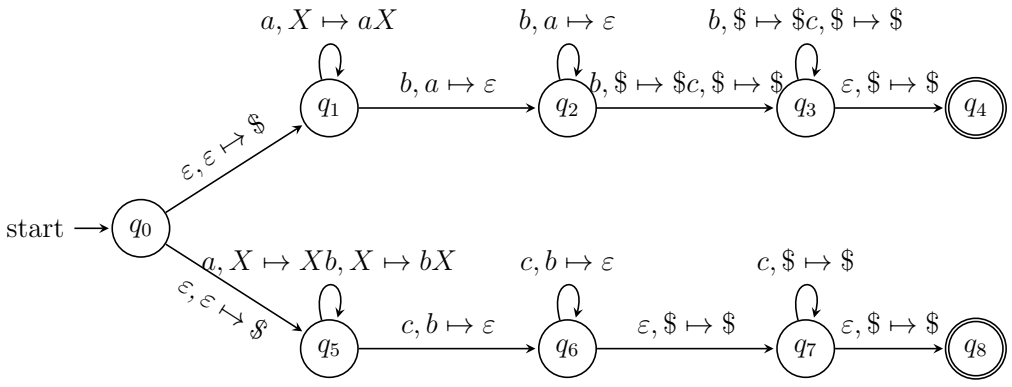

Figure 1: PDA per il linguaggio $\{a^i b^j c^k \mid i = j \text{ oppure } j = k\}$

Spiegazione del funzionamento

Il PDA opera nel seguente modo:

1. Inizia nello stato q_0 e sceglie non-deterministicamente uno dei due percorsi.
2. Nel primo percorso (verifica $i = j$):
   - Memorizza tutte le 'a' nello stack
   - Per ogni 'b' letta, rimuove una 'a' dallo stack
   - Se esattamente tutte le 'a' sono state rimosse quando iniziano le 'c', allora $i = j$
   - Accetta se dopo aver letto tutta la stringa, il percorso termina in q_4
3. Nel secondo percorso (verifica $j = k$):
   - Ignora tutte le 'a'
   - Memorizza tutte le 'b' nello stack
   - Per ogni 'c' letta, rimuove una 'b' dallo stack
   - Se esattamente tutte le 'b' sono state rimosse alla fine della stringa, allora $j = k$
   - Accetta se dopo aver letto tutta la stringa, il percorso termina in q_8

Esempi di accettazione

1. *aabbcc* - Accettata perché $i = j = 2$ e $j = k = 2$ (entrambe le condizioni sono soddisfatte)
2. *aaabbc* - Accettata perché $j = k = 2$ (seconda condizione soddisfatta)
3. *aabbccc* - Accettata perché $i = j = 2$ (prima condizione soddisfatta)
4. *aabbbcc* - Rifiutata perché $i = 2 \neq j = 3$ e $j = 3 \neq k = 2$ (nessuna condizione soddisfatta)

Esercizio 2

Costruisci un PDA per il linguaggio $\{ww^R \mid w \in \{0,1\}^*\}$, dove w^R indica la parola w scritta al contrario

Soluzione

Questo linguaggio rappresenta l'insieme delle stringhe palindrome di lunghezza pari, dove la prima metà è uguale al rovescio della seconda metà. Esempi di stringhe in questo linguaggio sono: ε, 00, 11, 0110, 1001, 010010, ecc.

Approccio

Per riconoscere questo linguaggio, il PDA deve:

1. Memorizzare la prima metà della stringa nello stack
2. Riconoscere in qualche modo il punto centrale della stringa
3. Verificare che la seconda metà sia il rovescio della prima metà, confrontando i simboli letti con quelli nello stack

Il punto cruciale è determinare quando siamo arrivati alla metà della stringa. Per risolvere questo problema, utilizzeremo un approccio non deterministico: l'automa indovinerà il punto centrale e poi verificherà se la sua scelta è corretta.

Definizione formale del PDA

Il PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ dove:

- $Q = \{q_0, q_1, q_2, q_3\}$ è l'insieme degli stati
- $\Sigma = \{0, 1\}$ è l'alfabeto di input
- $\Gamma = \{0, 1, \$\}$ è l'alfabeto dello stack
- q_0 è lo stato iniziale
- $F = \{q_3\}$ è l'insieme degli stati finali
- δ è la funzione di transizione definita come segue:

Funzione di transizione δ

- $\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$)\}$ (inizializza lo stack con il simbolo di fondo pila)
- $\delta(q_1, 0, X) = \{(q_1, 0X)\}$ per ogni $X \in \Gamma$ (memorizza '0' sullo stack)
- $\delta(q_1, 1, X) = \{(q_1, 1X)\}$ per ogni $X \in \Gamma$ (memorizza '1' sullo stack)
- $\delta(q_1, \varepsilon, X) = \{(q_2, X)\}$ per ogni $X \in \Gamma$ (indovina non-deterministicamente che siamo a metà)

- $\delta(q_2, 0, 0) = \{(q_2, \varepsilon)\}$ (confronta '0' con lo stack e rimuove se corrispondente)
- $\delta(q_2, 1, 1) = \{(q_2, \varepsilon)\}$ (confronta '1' con lo stack e rimuove se corrispondente)

- $\delta(q_2, \varepsilon, \$) = \{(q_3, \$)\}$ (accetta quando il confronto è completato e rimane solo il)

Rappresentazione grafica del PDA

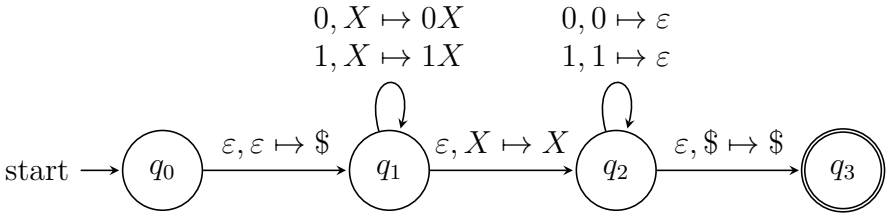

Figure 2: PDA per il linguaggio $\{ww^R \mid w \in \{0,1\}^*\}$

Spiegazione del funzionamento

Il PDA opera nel seguente modo:

1. Inizia nello stato q_0 e passa allo stato q_1 inserendo il simbolo di fondo pila \$.
2. Nello stato q_1, legge e memorizza la prima parte della stringa di input nello stack (push).
3. Ad un certo punto, l'automa "indovina" non-deterministicamente che ha raggiunto la metà della stringa e passa allo stato q_2.
4. Nello stato q_2, confronta ogni simbolo letto dalla seconda metà dell'input con il simbolo in cima allo stack (pop).
   - Se i simboli corrispondono, rimuove il simbolo dallo stack.
   - Se i simboli non corrispondono, il percorso di computazione fallisce.
5. Se alla fine dell'input lo stack contiene solo il simbolo di fondo pila \$, il PDA passa allo stato finale q_3 e accetta la stringa.

Grazie alla natura non deterministica del PDA, se esiste un modo di dividere la stringa in due parti tali che la seconda parte sia il rovescio della prima, il PDA accetterà la stringa.

Esempi di accettazione

1. 0110 - Accettata perché può essere divisa in $w = 01$ e $w^R = 10$
2. 1001 - Accettata perché può essere divisa in $w = 10$ e $w^R = 01$
3. 01010 - Rifiutata perché ha lunghezza dispari e non può essere della forma ww^R
4. 0011 - Rifiutata perché non può essere divisa in w e w^R (sarebbe $w = 00$ e $w^R = 00 \neq 11$)

Osservazioni

- La stringa vuota ε è accettata dal PDA, poiché può essere considerata come ww^R con $w = \varepsilon$.
- Il PDA è non deterministico perché deve "indovinare" il punto centrale della stringa.
- Un PDA deterministico per questo linguaggio richiederebbe un approccio differente e sarebbe più complesso.