Automi e Linguaggi Formali

Parte 6 – Automi a Pila

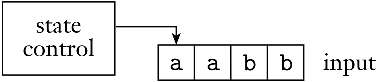
UNIVERSITÀ DEGLI STUDI DI PADOVA

Davide Bresolin
Ultimo aggiornamento: 31 marzo 2021

Sommario

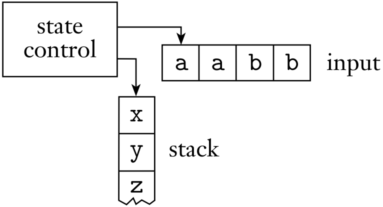
Dagli automi a stati finiti . . .

- Input: stringa di caratteri dell'alfabeto
- Memoria: stati
- Funzione di transizione: dato lo stato corrente ed un simbolo di input, stabilisce quali possono essere gli stati successivi



. . . agli Automi a Pila (PDA)

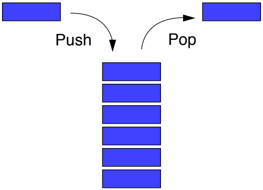
- Input: stringa di caratteri dell'alfabeto
- Memoria: stati + pila
- Funzione di transizione: dato lo stato corrente, un simbolo di input ed il simbolo in cima alla pila, stabilisce quali possono essere gli stati successivi e i simboli da scrivere sulla pila



Come funziona una pila

La pila è un dispositivo di memoria last in, first out (LIFO):

- Push: scrivi un nuovo simbolo in cima alla pila e "spingi giù" gli altri
- Pop: leggi e rimuovi il simbolo in cima alla pila (top)



La pila permette di avere memoria infinita (ad accesso limitato)

Esempio: $\{0^n 1^n \mid n \geq 0\}$

Un PDA usa la pila per contare 0 e 1:

- legge i simboli in input, e scrive ogni 0 letto sulla pila
- non appena vede gli 1, cancella uno 0 dalla pila per ogni 1 letto
- se l'input termina esattamente quando la pila si svuota, accetta
- se ci sono ancora 0 nella pila al termine dell'input, rifiuta
- se la pila si svuota prima della fine dell'input, rifiuta
- se qualche 0 compare nell'input dopo gli 1, rifiuta

Definizione di PDA

Un automa a pila (o Pushdown Automata, PDA) è una sestupla $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$:

- Q è l'insieme finito di stati
- Σ è l'alfabeto di input
- Γ è l'alfabeto della pila
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \mapsto 2^{Q \times \Gamma_\varepsilon}$ è la funzione di transizione
- $q_0 \in Q$ è lo stato iniziale
- $F \subseteq Q$ è l'insieme di stati accettanti

(dove $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ e $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$)

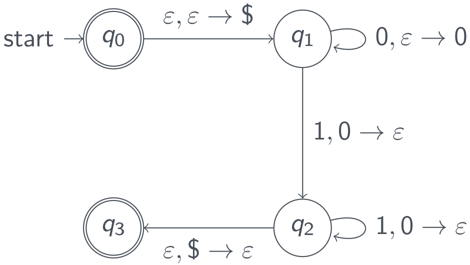
PDA per $\{0^n1^n \mid n \geq 0\}$

- $P = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{0, \$\}, \delta, q_0, \{q_0, q_3\})$
- con δ descritta dalla tabella:

Input:	0			1			ε		
Pila:	0	\$	ε	0	\$	ε	0	\$	ε
q_0									$\{(q_1, \$)\}$
q_1			$\{(q_1, 0)\}$	$\{(q_2, \varepsilon)\}$					
q_2				$\{(q_2, \varepsilon)\}$				$\{(q_3, \varepsilon)\}$	
q_3									

PDA per $\{0^n1^n \mid n \geq 0\}$

- $P = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{0, \$\}, \delta, q_0, \{q_0, q_3\})$
- con δ descritta dal diagramma di transizione:



Computazione e linguaggio di un PDA

Data una parola w, un PDA accetta la parola se:

- possiamo scrivere $w = w_1 w_2 \dots w_m$ dove $w_i \in \Sigma \cup \{\varepsilon\}$
- esistono una sequenza di stati $r_0, r_1, \dots, r_m \in Q$ e
- una sequenza di stringhe $s_0, s_1, s_2, \dots, s_m \in \Gamma^*$

tali che

1. $r_0 = q_0$ e $s_0 = \varepsilon$ (inizia dallo stato iniziale e pila vuota)
2. per ogni $i = 0, \dots, m-1$, $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$ con $s_i = at$ e $s_{i+1} = bt$ per qualche $a, b \in \Gamma_\varepsilon$ e $t \in \Gamma^*$ (rispetta la funzione di transizione)
3. $r_m \in F$ (la computazione termina in uno stato finale)

Esercizi

1. Costruisci un PDA per il linguaggio $\{a^i b^j c^k \mid i = j \text{ oppure } j = k\}$
2. Costruisci un PDA per il linguaggio $\{ww^R \mid w \in \{0,1\}^*\}$, dove w^R indica la parola w scritta al contrario

Accettazione per pila vuota

- la nostra definizione di PDA accetta le parole per stato finale
- esiste un altro modo per definire la condizione di accettazione:

Accettazione per pila vuota

Un PDA accetta la parola w per pila vuota se esiste una computazione che

- consuma tutto l'input
- termina con la pila vuota ($s_m = \varepsilon$)

Equivalenza

Per ogni linguaggio accettato da un PDA per stato finale esiste un PDA che accetta per pila vuota, e viceversa.

Sommario

Equivalenza con le grammatiche context-free

Theorem

Un linguaggio è context-free se solo se esiste un PDA che lo riconosce

- Sappiamo che un linguaggio è context-free se esiste una CFG che lo genera
- Mostreremo come trasformare la grammatica in un PDA
- E viceversa, come trasformare un PDA in una grammatica

Da CFG a PDA

Lemma

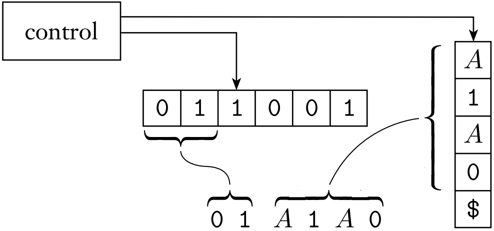
Se un linguaggio è context free, allora esiste un PDA che lo riconosce

Idea.

- Se L è context free, allora esiste una CFG G che lo genera
- Mostriamo come trasformare G in un PDA equivalente P
- P è fatto in modo da simulare i passi di derivazione di G
- P accetta w se esiste una derivazione di w in G

Rappresentare le stringhe intermedie

- Ogni derivazione è una sequenza di stringhe intermedie
- La pila memorizza le stringhe intermedie
- P trova le variabili nella stringa intermedia e fa le sostituzioni seguendo le regole di G
- Idea: metto nella pila solo i simboli dalla prima variabile in poi



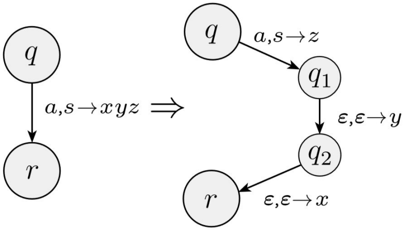
Definizione informale del PDA

1. Inserisci il simbolo marcatore \$ e la variabile iniziale S sulla pila
2. Ripeti i seguenti passi:
    1. Se la cima della pila è la variabile A: scegli una regola $A \to u$ e scrivi u sulla pila
    2. Se la cima della pila è un terminale a: leggi il prossimo simbolo di input.
        - se sono uguali, procedi
        - se sono diversi, rifiuta
    3. Se la cima della pila è \$: vai nello stato accettante

Notazione compatta

- supponiamo che il PDA vada da q a r quando legge a e fa il pop di s ...
- ... inserendo la stringa di tre caratteri $u = xyz$ sulla pila

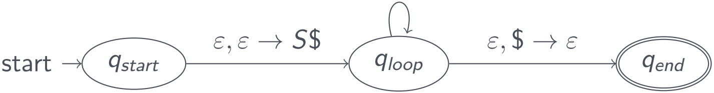
- per implementare il push multiplo dobbiamo aggiungere stati ausiliari



Data $G = (V, \Sigma, R, S)$, definiamo $P = (Q, \Sigma, \Gamma, q_{start}, F)$:

- $Q = \{q_{start}, q_{loop}, q_{end}\}$
- $\Gamma = \Sigma \cup V \cup \{\$\}$
- $F = \{q_{end}\}$
- funzione di transizione:

$$\begin{array}{ll} \varepsilon, A \to u & \text{per la regola } A \to u \\ a, a \to \varepsilon & \text{per il terminale } a \end{array}$$



Esempio

Trasformiamo la seguente CFG in PDA:

$$S \to aTb \mid b$$
$$T \to Ta \mid \varepsilon$$