Automi a pila

PushDown Automaton (PDA)

Un Automa a Pila è una sestupla $(Q, \Sigma, \Gamma, \delta, q0, F)$:

- Q è l'insieme finito degli stati dell'automa
- Σ è l'alfabeto dell'automa
- Γ è l'alfabeto dello stack (o pila) dell'automa
- $q_0 \in Q$ è lo stato iniziale dell'automa
- $F \subseteq Q$ è l'insieme degli stati accettanti dell'automa
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to P(Q \times \Gamma_\varepsilon)$ è la funzione di transizione dell'automa, dove
  se $(q, c) \in \delta(p, a, b)$ si ha che:
  - Viene letto il simbolo a dalla stringa in input e, se il simbolo b è in cima allo stack, allora l'automa passa dallo stato p allo stato q e il simbolo b viene sostituito dal simbolo c.
  - L'etichetta della transizione da p a q viene indicata come $a; b \to c$.

Dato $(q, c) \in \delta(p, a, b)$, dove δ è la funzione di transizione di un PDA, si ha che:

- Se $b, c = \varepsilon$ (dunque $a; \varepsilon \to \varepsilon$), l'automa leggerà a dalla stringa e passerà direttamente dallo stato p allo stato q, senza modificare lo stack.
- Se $b = \varepsilon$ e $c \neq \varepsilon$ (dunque $a; \varepsilon \to c$), l'automa leggerà a dalla stringa, passerà direttamente dallo stato p allo stato q e in cima allo stack viene aggiunto il simbolo c (**push**).
- Se $b \neq \varepsilon$ e $c = \varepsilon$ (dunque $a; b \to \varepsilon$), l'automa leggerà a e, se in cima allo stack vi è b, l'automa passerà dallo stato p allo stato q e rimuoverà b dalla cima dello stack (**pop**).

Stringa Accettata da un PDA

Sia $P := (Q, \Sigma, \Gamma, \delta, q_0, F)$ un PDA. Data una stringa $w := w_0 \dots w_k \in \Sigma^*$, dove $w_0, \dots, w_k \in \Sigma_\varepsilon$, si dice che w è accettata da P se esiste una sequenza di stati $r_0, r_1, \dots, r_{k+1} \in Q$ ed una sequenza di stringhe $s_0, s_1, \dots, s_n \in \Gamma^*$ tali che:

- $r_0 = q_0$
- $r_{k+1} \in F$
- $s_0 = \varepsilon$ (stack vuoto)
- $\forall i \in [0, k]$ si abbia che:
  - $(r_{i+1}, b) \in \delta(r_i, w_i, a)$
  - $s_i = at$
  - $s_{i+1} = bt$
    dove $a, b \in \Gamma_\varepsilon$ e $t \in \Gamma^*$ è la stringa composta dai caratteri nello stack.

Classe dei linguaggi riconosciuti da un PDA

Dato un alfabeto Σ, definiamo come classe dei linguaggi di Σ riconosciuti da un PDA il seguente insieme:

$$L(PDA) = \{L \subseteq \Sigma^* \mid \exists \text{ PDA } P \text{ t.c. } L = L(P)\}$$

Scrittura di una stringa sullo stack

Sia $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ un PDA. Dati $u_1, \dots, u_k \in \Gamma$, si introduce una notazione per cui δ possa ammettere la scrittura diretta sullo stack della stringa $u := u_1 \dots u_k$.
Ossia: $(q, u_1 \dots u_k) \in \delta(p, a, b) \Leftrightarrow \exists r_1, \dots, r_{k-1} \in Q$ tali che:

- $\delta(p, a, b) \ni (r_1, u_k)$
- $\delta(r_1, \varepsilon, \varepsilon) = \{(r_2, u_{k-1})\}$
- $\dots$
- $\delta(r_{k-1}, \varepsilon, \varepsilon) = \{(q, u_1)\}$

TEOREMA

> Un linguaggio è acontestuale (generato da una CFG) se e solo se esiste un PDA che lo riconosce. Per rendere più leggibile e chiara la dimostrazione del teorema, essa verrà scomposta in due lemmi, che rappresentano le due implicazioni della dimostrazione.
> Ossia, date le due classi dei linguaggi $\mathcal{L}(PDA)$ e CFL, si ha che:
>
> $$\mathcal{L}(PDA) = CFL$$

Prima Inclusione

Date le due classi dei linguaggi CFL e $\mathcal{L}(PDA)$, si ha che:

$$CFL \subseteq \mathcal{L}(PDA)$$

Ossia **ogni linguaggio context-free (CFL) può essere riconosciuto da un PDA**. L'idea chiave è **costruire un PDA a partire da una grammatica libera dal contesto (CFG)**.

Dimostrazione

Dato $L \in CFL$, sia $G = (V, \Sigma, R, S)$ la CFG tale che $L = L(G)$. Si considera il PDA $P = (Q, \Sigma, \Gamma, \delta, q_{start}, F)$ tale che:

- $Q = \{q_{start}, q_{loop}, q_{accept}\} \cup Q_\delta$, dove Q_δ sono i minimi stati aggiunti affinché la sua funzione δ sia ben definita.
- $\Gamma = V \cup \Sigma$ (lo stack contiene sia simboli non-terminali che terminali)
- $F = \{q_{accept}\}$ (stato finale)
- Dato $q_{start} \in Q$ si ha che

  $$\delta(q_{start}, \varepsilon, \varepsilon) = \{(q_{loop}, S\$)\}$$

  Lo stack parte con il simbolo iniziale S della CFG e un simbolo speciale `$` in fondo:

- $\forall A \in V$ si ha che

$$\delta(q_{loop}, \varepsilon, A) = \{(q_{loop}, u) \mid (A \to u) \in R,\ u \in \Gamma^*\}$$

  Quando il simbolo in cima allo stack è un non-terminale $A \in V$, il PDA può sostituirlo con qualsiasi destra della produzione $A \to u \in R$. Qui ε significa che il PDA **non legge un simbolo dall'input**, fa solo manipolazioni dello stack.
- $\forall a \in \Sigma$ si ha che

$$\delta(q_{loop}, a, a) = \{(q_{loop}, \varepsilon)\}$$

  Quando il simbolo in cima allo stack è un terminale $a \in \Sigma$, il PDA lo confronta con il simbolo corrente in input e lo consuma. Questo è il meccanismo che "legge" la stringa.
- Dato $q_{accept} \in Q$ si ha che

$$\delta(q_{loop}, \varepsilon, \$) = \{(q_{accept}, \varepsilon)\}$$

  Una volta che lo stack contiene solo il simbolo `$` speciale, il PDA può passare allo stato di accettazione
  - A questo punto, per costruzione stessa di P si ha che:

$$w \in L = L(G) \Leftrightarrow w \in L(P)$$

    dunque che $L = L(P) \in L(PDA)$.

□

Seconda Inclusione

Date le due classi dei linguaggi $\mathcal{L}(PDA)$ e CFL, si ha che:

$$\mathcal{L}(PDA) \subseteq CFL$$

1. Partiamo da un linguaggio accettato da un PDA

Sia: $L \in L(PDA), \quad P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ tale che $L = L(P)$
Obiettivo: costruire una **CFG** G tale che $L = L(G)$.

---

2. Normalizzazione del PDA

Costruiamo un PDA $P' = (Q', \Sigma, \Gamma, \delta', q_0, \{q_{accept}\})$ con le seguenti proprietà:

1. **Transizioni elementari:** ogni transizione effettua **solo push o pop**, mai sostituzioni complesse.

$$(q, c) \in \delta(p, a, b) \implies \exists r \in Q' \text{ t.c. } (r, \varepsilon) \in \delta'(p, a, b) \wedge \delta'(r, \varepsilon, \varepsilon) = \{(q, c)\}$$

2. **Stati aggiuntivi:**

$$Q' = Q \cup Q'_\delta \cup \{q_{accept}\}$$

3. **Accettazione:** unico stato accettante q_{accept}, con transizioni ε da ogni $q \in F$.
4. **Stack vuoto all'accettazione:** prima di accettare, lo stack deve essere vuoto.

Si ha quindi che per costruzione stessa di P' si ha che:

$$w \in L(P) \iff w \in L(P')$$

3. Costruzione della CFG $G = (V, \Sigma, R, S)$

- **Variabili:** $V = \{A_{p,q} \mid p, q \in Q'\}$
  Dove $A_{p,q}$ genera tutte le stringhe che portano P' dallo stato p allo stato q con stack vuoto.
- **Simbolo iniziale:** $S = A_{q_0, q_{accept}}$
- **Regole di produzione:**
  1. **Stack vuoto:** $A_{p,p} \to \varepsilon$
  2. **Transizioni push/pop di un simbolo intermedio**: Se $(r, u) \in \delta'(p, a, \varepsilon)$ e $(q, \varepsilon) \in \delta'(s, b, u)$, allora:

     $$A_{p,q} \to a A_{r,s} b$$

     Questo descrive che il PDA legge a, gestisce lo stack tramite u, e poi legge b.
  3. **Concatenazione di percorsi intermedi:** $A_{p,q} \to A_{p,r} A_{r,q}$. Questo permette di dividere il percorso da p a q in due segmenti intermedi.

---

4. Correttezza della CFG

Affermazione 1: Da derivazioni della CFG a computazioni del PDA

Enunciato:
Siano $p, q \in Q'$ e $x \in \Sigma^*$. Se nella CFG costruita dal PDA P' abbiamo:

$$A_{p,q} \overset{*}{\Longrightarrow} x$$

allora la stringa x **può essere letta dal PDA P' partendo dallo stato p e arrivando allo stato q, con lo stack completamente vuoto alla fine della computazione**.

Spiegazione dettagliata:

- $A_{p,q}$ è una variabile della CFG che rappresenta tutte le stringhe che permettono al PDA di andare da p a q senza lasciare simboli nello stack.
- La notazione $\overset{*}{\Longrightarrow}$ indica che x è derivabile da $A_{p,q}$ tramite zero o più produzioni della CFG.
- La costruzione della CFG assicura che ogni regola corrisponde esattamente a una sequenza di transizioni elementari del PDA:
  1. **Regola** $A_{p,q} \to a A_{r,s} b$:
     - Il PDA legge a in input, manipola lo stack per arrivare dallo stato r a s, e legge b per arrivare in q.
  2. **Regola** $A_{p,q} \to A_{p,r} A_{r,q}$:
     - Il PDA percorre un percorso intermedio passando da p a r e poi da r a q, sempre con stack vuoto alla fine.

Dimostrazione (idea):

- **Induzione sul numero di produzioni nella derivazione:**
  - **Caso base:** derivazione di lunghezza 1 ⇒ $A_{p,p} \to \varepsilon$. La stringa derivata è vuota, quindi il PDA resta nello stato p con stack vuoto.
  - **Passo induttivo:** derivazioni più lunghe: si divide la derivazione in pezzi corrispondenti alle produzioni CFG; ogni pezzo, per ipotesi induttiva, porta il PDA tra gli stati intermedi con stack vuoto. Combinando i pezzi, otteniamo che l'intera stringa x porta il PDA da p a q con stack vuoto.

Affermazione 2: Da computazioni del PDA a derivazioni della CFG

Enunciato:
Siano $p, q \in Q'$ e $x \in \Sigma^*$.
Se il PDA P' legge la stringa x partendo dallo stato p e arrivando nello stato q con lo stack completamente vuoto alla fine, allora:

$$A_{p,q} \overset{*}{\Longrightarrow} x$$

Spiegazione dettagliata:

- La CFG è costruita in modo da "catturare" tutte le possibili sequenze di transizioni del PDA.
- Ogni volta che il PDA legge un simbolo o manipola lo stack, esiste una regola della CFG che lo simula:
  1. **Transizione singola con push/pop:** corrisponde a una regola del tipo $A_{p,q} \to aA_{r,s}b$.
  2. **Percorso concatenato tra stati intermedi:** corrisponde a una regola del tipo $A_{p,q} \to A_{p,r}A_{r,q}$.

Dimostrazione (idea):

- **Induzione sul numero di transizioni percorse dal PDA:**
  - **Caso base:** zero transizioni ⇒ stringa vuota ⇒ $A_{p,p} \to \varepsilon$.
  - **Passo induttivo:** dividiamo la computazione del PDA in pezzi più piccoli:
    - Se lo stack viene riempito all'inizio e svuotato alla fine, la stringa si scompone come $x = ayb$, con $A_{p,q} \to aA_{r,s}b$ e $A_{r,s} \Rightarrow^* y$ per ipotesi induttiva.
    - Se lo stack si svuota durante la computazione, la stringa si scompone come $x = yz$, con $A_{p,r} \Rightarrow^* y$ e $A_{r,q} \Rightarrow^* z$, quindi $A_{p,q} \to A_{p,r}A_{r,q} \Rightarrow^* yz = x$.

---

5. Conclusione

Queste due affermazioni dimostrano la **corrispondenza biunivoca** tra:

- le stringhe generate dalla CFG G, e
- le stringhe lette dal PDA P' con stack vuoto alla fine.

Quindi:

$$x \in L(G) \iff x \in L(P')$$

Ma $L(P') = L(P)$, quindi:

$$L = L(P) = L(P') = L(G) \in CFL$$

$\Longrightarrow$ **Ogni linguaggio riconosciuto da un PDA è context-free.**

□