Esercizi aggiuntivi libro - Soluzioni

Linguaggi Regolari, Automi e Linguaggi Context-Free

Gabriel Rovesti
Corso di Laurea in Informatica - Università degli Studi di Padova

Aprile 2025

1 Esercizi sui Linguaggi Regolari

1.31 1. Per una stringa $w = w_1 w_2 \cdots w_n$, l'inversa di w, scritta w^R, è la stringa w in ordine inverso, $w_n \cdots w_2 w_1$. Per ogni linguaggio A, sia $A^R = \{w^R \mid w \in A\}$. Mostrare che se A è regolare, anche A^R è regolare.

Soluzione. Per dimostrare che A^R è regolare quando A è regolare, costruiremo un NFA che riconosce A^R a partire da un DFA che riconosce A.

Sia $M = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce il linguaggio regolare A. Costruiamo un NFA $M' = (Q, \Sigma, \delta', F, \{q_0\})$ che riconosce A^R come segue:

- Utilizziamo gli stessi stati Q di M
- Lo stato iniziale di M' è l'insieme degli stati finali F di M
- Gli stati finali di M' contengono solo lo stato iniziale $\{q_0\}$ di M
- La funzione di transizione δ' è l'inversa di δ:
  - $\delta'(q, a) = \{p \in Q \mid \delta(p, a) = q\}$ per ogni $q \in Q$ e $a \in \Sigma$

Intuitivamente, abbiamo invertito la direzione di tutte le transizioni del DFA originale e scambiato gli stati iniziali con quelli finali. Questo NFA accetta una stringa w^R se e solo se il DFA originale accetta w.

Formalmente, dimostriamo che $L(M') = A^R$:

- Se $w^R \in A^R$, allora $w \in A$, quindi $\delta^*(q_0, w) \in F$ nel DFA originale M. Possiamo dimostrare per induzione sulla lunghezza di w che nell'NFA M', $q_0 \in \delta'^*(F, w^R)$. Quindi w^R è accettata da M'.
- Se w^R è accettata da M', allora $q_0 \in \delta'^*(F, w^R)$. Usando lo stesso ragionamento, possiamo dimostrare che $\delta^*(q_0, w) \in F$ nel DFA originale M, quindi $w \in A$ e $w^R \in A^R$.

Poiché gli NFA riconoscono esattamente i linguaggi regolari, e M' è un NFA, A^R è un linguaggio regolare quando A è regolare.

1.38 2. Un all-NFA M è una 5-tupla $(Q, \Sigma, \delta, q_0, F)$ che accetta $x \in \Sigma^*$ se ogni possibile stato in cui M potrebbe trovarsi dopo aver letto l'input x è uno stato in F. Si noti che, in contrasto, un NFA ordinario accetta una stringa se almeno uno stato tra questi possibili stati è uno stato di accettazione. Dimostrare che gli all-NFA riconoscono la classe dei linguaggi regolari.

Soluzione. Per dimostrare che gli all-NFA riconoscono esattamente la classe dei linguaggi regolari, mostreremo due direzioni:

Parte 1: Ogni linguaggio riconosciuto da un all-NFA è regolare.

Sia $M = (Q, \Sigma, \delta, q_0, F)$ un all-NFA che riconosce il linguaggio $L(M)$. Definiamo un DFA $M' = (Q', \Sigma, \delta', q_0', F')$ come segue:

- $Q' = \mathcal{P}(Q)$ (insieme delle parti di Q)
- $q_0' = \{q_0\}$
- $\delta'(S, a) = \bigcup_{q \in S} \delta(q, a)$ per ogni $S \subseteq Q$ e $a \in \Sigma$
- $F' = \{S \in Q' \mid S \subseteq F\}$ (tutti i sottoinsiemi di Q i cui elementi sono tutti in F)

Il DFA M' simula l'all-NFA M tenendo traccia dell'insieme di tutti i possibili stati in cui M potrebbe trovarsi. M' accetta una stringa w se e solo se ogni possibile stato in cui M potrebbe trovarsi dopo aver letto w è uno stato di accettazione in M, cioè se e solo se $\delta'^*(q_0', w) \subseteq F$.

Poiché M' è un DFA, il linguaggio che riconosce è regolare. Quindi, ogni linguaggio riconosciuto da un all-NFA è regolare.

Parte 2: Ogni linguaggio regolare può essere riconosciuto da un all-NFA.

Sia L un linguaggio regolare. Allora esiste un DFA $M = (Q, \Sigma, \delta, q_0, F)$ che riconosce L. Possiamo costruire un all-NFA $M' = (Q, \Sigma, \delta', q_0, F')$ che riconosce lo stesso linguaggio come segue:

- $\delta'(q, a) = \{\delta(q, a)\}$ per ogni $q \in Q$ e $a \in \Sigma$ (ogni transizione porta a esattamente uno stato, come nel DFA)
- $F' = F$ (stessi stati di accettazione)

Poiché M' è deterministico (ogni transizione porta a esattamente uno stato), per ogni stringa w, c'è esattamente un possibile stato in cui M' potrebbe trovarsi dopo aver letto w. Quindi, M' accetta w se e solo se questo unico stato è in F', che è esattamente la condizione per cui M accetta w.

Quindi, ogni linguaggio regolare può essere riconosciuto da un all-NFA.

Combinando le due parti, concludiamo che gli all-NFA riconoscono esattamente la classe dei linguaggi regolari.

1.40 3. Ricordiamo che una stringa x è un prefisso di una stringa y se esiste una stringa z tale che $xz = y$, e che x è un prefisso proprio di y se inoltre $x \neq y$. In ognuna delle seguenti parti, definiamo un'operazione su un linguaggio A. Mostrare che la classe dei linguaggi regolari è chiusa sotto tale operazione.

(a) $NOPREFIX(A) = \{w \in A \mid \text{nessun prefisso proprio di } w \text{ è un membro di } A\}$.

(b) $NOEXTEND(A) = \{w \in A \mid w \text{ non è il prefisso proprio di alcuna stringa in } A\}$.

Soluzione. **(a)** Dimostriamo che se A è un linguaggio regolare, allora $NOPREFIX(A)$ è regolare.

Sia A un linguaggio regolare. Allora esiste un DFA $M = (Q, \Sigma, \delta, q_0, F)$ che riconosce A. Costruiamo un DFA $M' = (Q', \Sigma, \delta', q_0', F')$ che riconosce $NOPREFIX(A)$ come segue:

- $Q' = Q \times \{0, 1\}$, dove il secondo componente è un flag che indica se abbiamo incontrato un prefisso proprio che è in A
- $q_0' = (q_0, 0)$
- $\delta'((q, b), a) = (\delta(q, a), b')$, dove $b' = 1$ se $b = 1$ o $\delta(q, a) \in F$, altrimenti $b' = 0$
- $F' = \{(q, 0) \mid q \in F\}$

Il DFA M' simula M e tiene traccia se ha incontrato un prefisso proprio della stringa corrente che è in A. Accetta solo se lo stato finale è in F e non ha incontrato alcun prefisso proprio che è in A.

Dimostriamo che $L(M') = NOPREFIX(A)$:

- Se $w \in NOPREFIX(A)$, allora $w \in A$ e nessun prefisso proprio di w è in A. Quindi, $\delta^*(q_0, w) \in F$ nel DFA originale M, e durante la simulazione di M' su w, il flag rimane 0 fino all'ultimo carattere, quando lo stato finale è $(q, 0)$ con $q \in F$. Quindi w è accettata da M'.
- Se w è accettata da M', allora lo stato finale è $(q, 0)$ con $q \in F$. Ciò significa che $\delta^*(q_0, w) \in F$ nel DFA originale M, quindi $w \in A$. Inoltre, il flag è 0, quindi nessun prefisso proprio di w è in A. Quindi, $w \in NOPREFIX(A)$.

Poiché M' è un DFA, $NOPREFIX(A)$ è un linguaggio regolare quando A è regolare.

(b) Dimostriamo che se A è un linguaggio regolare, allora $NOEXTEND(A)$ è regolare.

Sia A un linguaggio regolare. Allora esiste un DFA $M = (Q, \Sigma, \delta, q_0, F)$ che riconosce A. Definiamo

- $PREFIX(A) = \{w \mid \exists x \in \Sigma^+ \text{ tale che } wx \in A\}$

È noto che $PREFIX(A)$ è regolare quando A è regolare. Quindi, $NOEXTEND(A) = A \setminus PREFIX(A)$ è regolare, perché i linguaggi regolari sono chiusi sotto differenza di insiemi.

Alternativamente, possiamo costruire direttamente un DFA per $NOEXTEND(A)$. Sia $M = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce A. Costruiamo un DFA $M' = (Q, \Sigma, \delta, q_0, F')$ che riconosce $NOEXTEND(A)$ come segue:

- $F' = \{q \in F \mid \delta(q, a) \notin F \text{ per ogni } a \in \Sigma\}$

Cioè, F' consiste degli stati finali di M da cui non è possibile raggiungere un altro stato finale leggendo un singolo simbolo. Questo assicura che se una stringa w termina in uno stato in F', allora w non è il prefisso proprio di alcuna stringa in A.

Dimostriamo che $L(M') = NOEXTEND(A)$:

- Se $w \in NOEXTEND(A)$, allora $w \in A$ e non esiste $x \in \Sigma^+$ tale che $wx \in A$. Quindi, $\delta^*(q_0, w) \in F$ nel DFA originale M, e da questo stato non è possibile raggiungere un altro stato finale leggendo qualsiasi sequenza di simboli. In particolare, $\delta(\delta^*(q_0, w), a) \notin F$ per ogni $a \in \Sigma$, quindi $\delta^*(q_0, w) \in F'$. Quindi w è accettata da M'.

- Se w è accettata da M', allora $\delta^*(q_0, w) \in F'$. Ciò significa che $\delta^*(q_0, w) \in F$ nel DFA originale M, quindi $w \in A$. Inoltre, $\delta(\delta^*(q_0, w), a) \notin F$ per ogni $a \in \Sigma$, quindi non esiste $a \in \Sigma$ tale che $wa \in A$. Per induzione, non esiste $x \in \Sigma^+$ tale che $wx \in A$. Quindi, $w \in NOEXTEND(A)$.

Poiché M' è un DFA, $NOEXTEND(A)$ è un linguaggio regolare quando A è regolare.

1.41 4. Per linguaggi A e B, il perfect shuffle di A e B è il linguaggio

$$S = \{w \mid w = a_1b_1a_2b_2 \dots a_kb_k, \text{ dove } a_1a_2 \dots a_k \in A \text{ e } b_1b_2 \dots b_k \in B, \text{ ogni } a_i, b_i \in \Sigma\}$$

Mostrare che la classe dei linguaggi regolari è chiusa rispetto al perfect shuffle.

Soluzione. Per dimostrare che il perfect shuffle di due linguaggi regolari è regolare, costruiremo un automa che riconosce tale linguaggio a partire dagli automi che riconoscono A e B.

Siano $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ e $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ due DFA che riconoscono rispettivamente A e B.

Costruiamo un nuovo NFA $M = (Q, \Sigma, \delta, q_0, F)$ che riconosce il perfect shuffle di A e B come segue:

- $Q = Q_A \times Q_B \times \{0, 1\}$, dove il terzo componente indica se il prossimo simbolo dovrebbe essere da A (valore 0) o da B (valore 1)
- $q_0 = (q_{0A}, q_{0B}, 0)$
- $F = \{(q_A, q_B, 1) \mid q_A \in F_A \text{ e } q_B \in F_B\}$
- La funzione di transizione δ è definita come:
  - $\delta((q_A, q_B, 0), a) = \{(\delta_A(q_A, a), q_B, 1)\}$ per ogni $a \in \Sigma$
  - $\delta((q_A, q_B, 1), b) = \{(q_A, \delta_B(q_B, b), 0)\}$ per ogni $b \in \Sigma$

L'automa M funziona alternando la lettura di simboli per A e B. Inizia leggendo un simbolo per A (stato con terzo componente 0), poi passa a leggere un simbolo per B (stato con terzo componente 1), e così via.

L'automa accetta solo se, dopo aver letto un numero pari di simboli (l'ultimo da B), entrambi gli stati per A e B sono di accettazione.

Formalmente, dimostriamo che $L(M)$ è esattamente il perfect shuffle di A e B:

- Se $w = a_1b_1a_2b_2\dots a_kb_k$ è nel perfect shuffle di A e B, allora $a_1a_2\dots a_k \in A$ e $b_1b_2\dots b_k \in B$. Quindi, $\delta_A^*(q_{0A}, a_1a_2\dots a_k) \in F_A$ e $\delta_B^*(q_{0B}, b_1b_2\dots b_k) \in F_B$. Durante la simulazione di M su w, alterniamo tra leggere simboli per A e B, e alla fine raggiungiamo lo stato $(\delta_A^*(q_{0A}, a_1a_2\dots a_k), \delta_B^*(q_{0B}, b_1b_2\dots b_k), 1)$, che è in F. Quindi w è accettata da M.

- Se w è accettata da M, allora w ha un numero pari di simboli e, dopo aver letto w, M si trova in uno stato $(q_A, q_B, 1)$ con $q_A \in F_A$ e $q_B \in F_B$. Ciò significa che w può essere scritto come $w = a_1b_1a_2b_2\dots a_kb_k$, dove $\delta_A^*(q_{0A}, a_1a_2\dots a_k) \in F_A$ e $\delta_B^*(q_{0B}, b_1b_2\dots b_k) \in F_B$. Quindi, $a_1a_2\dots a_k \in A$ e $b_1b_2\dots b_k \in B$, e w è nel perfect shuffle di A e B.

Poiché gli NFA riconoscono esattamente i linguaggi regolari, e M è un NFA, il perfect shuffle di A e B è un linguaggio regolare quando A e B sono regolari.

1.42 5. Per linguaggi A e B, lo shuffle di A e B è il linguaggio

$$S = \{w \mid w = a_1b_1\dots a_kb_k,\ \text{dove } a_1\dots a_k \in A \text{ e } b_1\dots b_k \in B,\ \text{ogni } a_i, b_i \in \Sigma^*\}$$

Mostrare che la classe dei linguaggi regolari è chiusa rispetto allo shuffle.

Soluzione. A differenza del perfect shuffle, dove ogni a_i e b_i sono singoli simboli, nello shuffle generale possono essere stringhe di lunghezza arbitraria (inclusa la stringa vuota). Dimostriamo che lo shuffle di due linguaggi regolari è ancora regolare.

Siano $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ e $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ due DFA che riconoscono rispettivamente A e B.

Costruiamo un NFA $M = (Q, \Sigma, \delta, q_0, F)$ che riconosce lo shuffle di A e B come segue:

- $Q = Q_A \times Q_B \times \{0, 1, 2\}$, dove il terzo componente indica il modo in cui stiamo processando la stringa:
  - 0: Possiamo scegliere di iniziare a leggere per A o per B
  - 1: Stiamo leggendo per A
  - 2: Stiamo leggendo per B
- $q_0 = (q_{0A}, q_{0B}, 0)$
- $F = \{(q_A, q_B, m) \mid q_A \in F_A \text{ e } q_B \in F_B \text{ e } m \in \{0, 1, 2\}\}$
- La funzione di transizione δ è definita come:
  - $\delta((q_A, q_B, 0), a) = \{(\delta_A(q_A, a), q_B, 1), (q_A, \delta_B(q_B, a), 2)\}$ per ogni $a \in \Sigma$
  - $\delta((q_A, q_B, 0), \varepsilon) = \{(q_A, q_B, 1), (q_A, q_B, 2)\}$ (transizioni ε per scegliere il modo)
  - $\delta((q_A, q_B, 1), a) = \{(\delta_A(q_A, a), q_B, 1)\}$ per ogni $a \in \Sigma$
  - $\delta((q_A, q_B, 1), \varepsilon) = \{(q_A, q_B, 2)\}$ (transizione ε per passare da A a B)
  - $\delta((q_A, q_B, 2), a) = \{(q_A, \delta_B(q_B, a), 2)\}$ per ogni $a \in \Sigma$
  - $\delta((q_A, q_B, 2), \varepsilon) = \{(q_A, q_B, 1)\}$ (transizione ε per passare da B a A)

L'NFA M funziona simulando simultaneamente i DFA M_A e M_B e alternando tra di essi in modo non deterministico. Può scegliere di leggere un segmento di input per A, poi un segmento per B, e così via, fino a che entrambi i DFA raggiungono stati di accettazione.

Per dimostrare formalmente che $L(M)$ è esattamente lo shuffle di A e B, consideriamo:

- Se $w = a_1 b_1 \dots a_k b_k$ è nello shuffle di A e B, dove $a_1 \dots a_k \in A$ e $b_1 \dots b_k \in B$, allora M può simulare la lettura di a_1 nel DFA M_A, poi la lettura di b_1 nel DFA M_B, e così via. Alla fine, entrambi i DFA saranno in stati di accettazione, quindi M accetta w.
- Se w è accettata da M, allora esiste un percorso di computazione in cui M alterna tra la lettura per A e per B, e alla fine entrambi i DFA sono in stati di accettazione. Ciò significa che w può essere decomposto come $w = a_1 b_1 \dots a_k b_k$, dove $a_1 \dots a_k \in A$ e $b_1 \dots b_k \in B$. Quindi, w è nello shuffle di A e B.

Poiché gli NFA riconoscono esattamente i linguaggi regolari, e M è un NFA, lo shuffle di A e B è un linguaggio regolare quando A e B sono regolari.

1.70 6. Definiamo l'operazione avoids per i linguaggi A e B come

$$A \text{ avoids } B = \{w \mid w \in A \text{ e } w \text{ non contiene alcuna stringa in } B \text{ come sottostringa}\}$$

Dimostrare che la classe dei linguaggi regolari è chiusa rispetto all'operazione avoids.

Soluzione. Per dimostrare che la classe dei linguaggi regolari è chiusa rispetto all'operazione avoids, mostreremo che se A e B sono linguaggi regolari, allora A avoids B è regolare.

Sia A un linguaggio regolare e B un linguaggio regolare. Definiamo:

$$C = \Sigma^* B \Sigma^* = \{w \mid w \text{ contiene una sottostringa in } B\}$$

È noto che se B è regolare, allora anche C è regolare. Infatti, se $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ è un DFA che riconosce B, possiamo costruire un NFA M_C che riconosce C aggiungendo transizioni ε dallo stato iniziale a tutti gli stati, e da tutti gli stati finali a tutti gli stati.

Ora, A avoids $B = A \setminus C = A \cap \overline{C}$. Poiché i linguaggi regolari sono chiusi sotto complemento e intersezione, A avoids B è regolare quando A e B sono regolari.

Alternativamente, possiamo costruire direttamente un DFA per A avoids B:

Sia $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ un DFA che riconosce A e $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ un DFA che riconosce B. Costruiamo un DFA $M = (Q, \Sigma, \delta, q_0, F)$ che riconosce A avoids B come segue:

- $Q = Q_A \times \mathcal{P}(Q_B)$
- $q_0 = (q_{0A}, \{q_{0B}\})$
- $F = \{(q_A, S) \mid q_A \in F_A \text{ e } S \cap F_B = \emptyset\}$
- La funzione di transizione δ è definita come:
  - $\delta((q_A, S), a) = (\delta_A(q_A, a), S')$, dove $S' = \{\delta_B(q, a) \mid q \in S\} \cup \{q_{0B}\}$

Il DFA M simula simultaneamente il DFA M_A e tutti i possibili "stati attivi" del DFA M_B. Per ogni simbolo letto, aggiorniamo lo stato in M_A e tutti gli stati attivi in M_B, e aggiungiamo anche lo stato iniziale di M_B per verificare se una nuova occorrenza di una stringa in B inizia in quella posizione.

M accetta una stringa w se e solo se $w \in A$ (lo stato in M_A è di accettazione) e w non contiene alcuna sottostringa in B (nessuno degli stati attivi in M_B è di accettazione).

Dimostriamo che $L(M) = A$ avoids B:

- Se $w \in A$ avoids B, allora $w \in A$ e w non contiene alcuna sottostringa in B. Quindi, $\delta_A^*(q_{0A}, w) \in F_A$ e, durante la simulazione di M su w, nessuno degli stati attivi in M_B raggiunge mai uno stato in F_B. Quindi, lo stato finale in M è (q_A, S) con $q_A \in F_A$ e $S \cap F_B = \emptyset$, cioè è in F. Quindi w è accettata da M.

- Se w è accettata da M, allora lo stato finale in M è (q_A, S) con $q_A \in F_A$ e $S \cap F_B = \emptyset$. Ciò significa che $\delta_A^*(q_{0A}, w) \in F_A$, quindi $w \in A$. Inoltre, nessuno degli stati attivi in M_B è di accettazione, quindi w non contiene alcuna sottostringa in B. Quindi, $w \in A$ avoids B.

Poiché M è un DFA, A avoids B è un linguaggio regolare quando A e B sono regolari.

2 Esercizi sui Linguaggi Context-Free

2.43 7. Per stringhe w e t, scriviamo $w \cong t$ se i simboli di w sono una permutazione dei simboli di t. In altre parole, $w \cong t$ se t e w hanno gli stessi simboli nelle stesse quantità, ma possibilmente in un ordine diverso.

Per ogni stringa w, definiamo $SCRAMBLE(w) = \{t \mid t \cong w\}$. Per ogni linguaggio A, sia $SCRAMBLE(A) = \{t \mid t \in SCRAMBLE(w)$ per qualche $w \in A\}$.

(a) Mostrare che se $\Sigma = \{0, 1\}$, allora lo $SCRAMBLE$ di un linguaggio regolare è context-free.

(b) Cosa accade nella parte (a) se Σ contiene tre o più simboli? Dimostrare la risposta.

Soluzione. **(a)** Dimostriamo che se $\Sigma = \{0, 1\}$, allora lo $SCRAMBLE$ di un linguaggio regolare è context-free.

Sia A un linguaggio regolare su $\Sigma = \{0, 1\}$. Per ogni stringa $w \in A$, $SCRAMBLE(w)$ dipende solo dal numero di 0 e 1 in w. Definiamo:

$$SCRAMBLE(A) = \{t \mid t \text{ ha lo stesso numero di 0 e 1 di qualche stringa in } A\}$$

Per costruire una CFG che genera $SCRAMBLE(A)$, osserviamo che possiamo partire da un DFA $M = (Q, \Sigma, \delta, q_0, F)$ che riconosce A e costruire un CFG che genera stringhe con lo stesso numero di 0 e 1 di qualche stringa accettata da M.

Costruiamo una CFG $G = (V, \Sigma, R, S)$ come segue:

- $V = \{S\} \cup \{A_{i,j,q} \mid 0 \leq i, j \leq n, q \in Q\}$, dove n è una costante sufficientemente grande (maggiore della lunghezza del pumping lemma per A)

- Le regole di produzione R includono:

  - $S \to A_{0,0,q_0}$
  - $A_{i,j,q} \to 0A_{i+1,j,\delta(q,0)} \mid 1A_{i,j+1,\delta(q,1)}$ per ogni $i, j < n$ e $q \in Q$
  - $A_{i,j,q} \to 0^i 1^j$ per ogni $q \in F$

Questa CFG genera stringhe con lo stesso numero di 0 e 1 di qualche stringa accettata da M, ma permette qualsiasi ordinamento dei simboli. Quindi, $L(G) = SCRAMBLE(A)$.

Possiamo perfezionare questa costruzione per gestire linguaggi regolari infiniti usando il pumping lemma e la proprietà che per ogni linguaggio regolare esiste un insieme finito di "parole rappresentative" tale che ogni stringa nel linguaggio è una variante di una di queste parole con alcune porzioni ripetute. Per ciascuna di queste parole rappresentative, possiamo costruire una produzione che genera stringhe con lo stesso numero di 0 e 1, ma in qualsiasi ordine.

Quindi, se $\Sigma = \{0, 1\}$, lo $SCRAMBLE$ di un linguaggio regolare è context-free.

(b) Mostriamo che esiste un linguaggio regolare A su $\Sigma = \{0, 1, 2\}$ tale che

$$SCRAMBLE(A) = \bigcup_{n,m \geq 0} \{w \in \{0, 1, 2\}^* \mid |w|_0 = n + m,\ |w|_1 = n,\ |w|_2 = m\}$$

non sia context-free.

Definiamo

$$A = (01)^*(02)^*.$$

Notiamo che:

- $(01)^*$ è regolare;
- $(02)^*$ è regolare;
- Per chiusura rispetto alla concatenazione, A è regolare.

Ora, per ogni $w \in A$, si ha che

$$w = (01)^n(02)^m,$$

per qualche $n, m \geq 0$. Tale stringa contiene:

- Dal fattore $(01)^n$: esattamente n occorrenze di 0 e n occorrenze di 1.
- Dal fattore $(02)^m$: esattamente m occorrenze di 0 e m occorrenze di 2.

Quindi, complessivamente,

$$|w|_0 = n + m, \quad |w|_1 = n, \quad |w|_2 = m.$$

Pertanto, lo $SCRAMBLE$ di w è l'insieme di tutte le stringhe in $\{0, 1, 2\}^*$ che contengono esattamente $n + m$ occorrenze di 0, n occorrenze di 1 e m occorrenze di 2. In altre parole,

$$SCRAMBLE(A) = \{w \in \{0, 1, 2\}^* \mid |w|_0 = |w|_1 + |w|_2\}.$$

È noto (e si può dimostrare tramite il pumping lemma per le CFL) che il linguaggio

$$L = \{w \in \{0,1,2\}^* \mid |w|_0 = |w|_1 + |w|_2\}$$

non è context-free.

Quindi, poiché

$$SCRAMBLE(A) = L,$$

abbiamo dimostrato che per Σ contenente almeno tre simboli lo $SCRAMBLE$ di un linguaggio regolare (in questo caso $A = (01)^*(02)^*$) può non essere context-free.

2.44 8. Se A e B sono linguaggi, definiamo $A \circ B = \{xy \mid x \in A \text{ e } y \in B \text{ e } |x| = |y|\}$. Mostrare che se A e B sono linguaggi regolari, allora $A \circ B$ è un linguaggio context-free.

Soluzione. Per dimostrare che $A \circ B$ è un linguaggio context-free quando A e B sono linguaggi regolari, costruiremo una grammatica context-free che genera $A \circ B$.

Siano $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ e $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ due DFA che riconoscono rispettivamente A e B.

Costruiamo una grammatica context-free $G = (V, \Sigma, R, S)$ che genera $A \circ B$:

- $V = \{S\} \cup \{[p,q] \mid p \in Q_A, q \in Q_B\}$
- Le regole di produzione R includono:
  - $S \to [q_{0A}, q_{0B}]$
  - $[p,q] \to a[p',q']b$ per ogni $p,p' \in Q_A$, $q,q' \in Q_B$, $a,b \in \Sigma$ tali che $\delta_A(p,a) = p'$ e $\delta_B(q,b) = q'$
  - $[p,q] \to \varepsilon$ per ogni $p \in F_A$ e $q \in F_B$

Questa grammatica funziona nel modo seguente:

- Il non terminale $[p,q]$ rappresenta che siamo nello stato p del DFA M_A e nello stato q del DFA M_B.
- La produzione $[p,q] \to a[p',q']b$ genera simultaneamente un simbolo a per la prima metà della stringa (che deve essere in A) e un simbolo b per la seconda metà (che deve essere in B).
- La produzione $[p,q] \to \varepsilon$ termina la generazione quando entrambi i DFA sono in stati di accettazione.

Dimostriamo che $L(G) = A \circ B$:

- Se $w \in A \circ B$, allora $w = xy$ con $x \in A$, $y \in B$ e $|x| = |y|$. Possiamo generare w in G partendo da S, derivando il non terminale $[q_{0A}, q_{0B}]$, e poi applicando le produzioni $[p,q] \to a[p',q']b$ per generare simultaneamente x e y. Poiché $x \in A$ e $y \in B$, i DFA M_A e M_B terminano in stati di accettazione, quindi possiamo applicare la produzione $[p,q] \to \varepsilon$ alla fine. Quindi, $w \in L(G)$.

- Se $w \in L(G)$, allora w può essere generato in G partendo da S. Durante la derivazione, generiamo alternativamente simboli per la prima e la seconda metà della stringa, simulando i DFA M_A e M_B. Alla fine, raggiungiamo un non terminale $[p, q]$ con $p \in F_A$ e $q \in F_B$, che deriviamo in ε. Quindi, $w = xy$ con $x \in A$, $y \in B$ e $|x| = |y|$, cioè $w \in A \circ B$.

Poiché G è una grammatica context-free, $A \circ B$ è un linguaggio context-free quando A e B sono linguaggi regolari.

2.49 9. Definiamo la rotational closure di un linguaggio A come $RC(A) = \{yx \mid xy \in A\}$. Mostrare che la classe dei linguaggi context-free è chiusa rispetto alla rotational closure.

Soluzione. Per dimostrare che la classe dei linguaggi context-free è chiusa rispetto alla rotational closure, mostreremo che se A è un linguaggio context-free, allora $RC(A) = \{yx \mid xy \in A\}$ è anch'esso un linguaggio context-free.

Sia A un linguaggio context-free. Allora esiste una grammatica context-free $G = (V, \Sigma, R, S)$ tale che $L(G) = A$. Costruiamo una grammatica context-free $G' = (V', \Sigma, R', S')$ che genera $RC(A)$.

G' è definita come segue:

- $V' = V \cup \{S'\} \cup \{[X, a] \mid X \in V, a \in \Sigma\}$, dove S' è un nuovo simbolo non terminale e $[X, a]$ sono nuovi simboli non terminali per ogni $X \in V$ e $a \in \Sigma$

- Le regole di produzione R' includono:

  - $S' \to S$ (per generare stringhe in A che non vengono ruotate)
  - $S' \to aB$ per ogni produzione $S \to Ba$ in R (per generare rotazioni che cominciano con l'ultimo simbolo)
  - $S' \to [X, a]Y$ per ogni produzione $S \to XY$ in R e ogni $a \in \Sigma$ (per iniziare a tenere traccia di dove siamo nella derivazione)
  - $[X, a] \to b[Y, a]$ per ogni produzione $X \to bY$ in R (per propagare il simbolo da ruotare)
  - $[X, a] \to b$ per ogni produzione $X \to ba$ in R (per completare la rotazione)

La grammatica G' funziona generando stringhe in due modi:

- Usando la regola $S' \to S$ seguita dalle regole di G, G' può generare tutte le stringhe in A.

- Usando le altre regole, G' può generare tutte le rotazioni delle stringhe in A.

Tuttavia, questa costruzione potrebbe essere complicata e non catturare tutte le possibili rotazioni. Una costruzione alternativa utilizza un Push-Down Automaton (PDA).

Sia $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ un PDA che accetta A per stato finale. Costruiamo un nuovo PDA $M' = (Q', \Sigma, \Gamma', \delta', q_0', Z_0', F')$ che accetta $RC(A)$:

- $Q' = Q \cup \{q_0', q_1, q_2\}$, dove q_0', q_1, q_2 sono nuovi stati

- $\Gamma' = \Gamma \cup \{Z_0', \#\}$, dove $Z_0', \#$ sono nuovi simboli di stack

- $F' = F$
- Le transizioni in δ' includono:
  - $\delta'(q_0', \varepsilon, Z_0') = \{(q_1, Z_0\#)\}$ (inizializziamo lo stack con Z_0 seguito da un marcatore)
  - $\delta'(q_1, a, Z) = \{(q_1, aZ)\}$ per ogni $a \in \Sigma$ e $Z \in \Gamma'$ (memorizziamo il prefisso sullo stack)
  - $\delta'(q_1, \varepsilon, Z) = \{(q_2, Z)\}$ per ogni $Z \in \Gamma'$ (passiamo alla fase di lettura del suffisso)
  - $\delta'(q_2, a, Z) = \{(q_2, Z)\}$ per ogni $a \in \Sigma$ e $Z \in \Gamma \cup \{Z_0\}$ (leggiamo il suffisso senza modificare lo stack)
  - $\delta'(q_2, \varepsilon, \#) = \{(q_0, \varepsilon)\}$ (quando raggiungiamo il marcatore, passiamo allo stato iniziale del PDA originale)
  - $\delta'(q_2, \varepsilon, a) = \{(q_2, \varepsilon)\}$ per ogni $a \in \Sigma$ (popping dei simboli del prefisso dallo stack)
  - Tutte le transizioni di δ (il PDA originale inizia a processare dall'inizio dopo aver letto il suffisso e il prefisso)

Questo PDA funziona in tre fasi:

- Nella prima fase, memorizza il prefisso della stringa sullo stack.
- Nella seconda fase, legge il suffisso della stringa e poi svuota lo stack (il prefisso).
- Nella terza fase, simula il PDA originale su tutta la stringa (suffisso seguito dal prefisso).

Questo PDA accetta una stringa yx se e solo se $xy \in A$, cioè accetta esattamente $RC(A)$.

Poiché i linguaggi accettati dai PDA sono esattamente i linguaggi context-free, e M' accetta $RC(A)$, concludiamo che $RC(A)$ è un linguaggio context-free quando A è un linguaggio context-free.