Gemini 1

🎓 Simulazione Secondo Parziale - PROVA 6 (Obsidian Mode)

Istruzioni: Prova a risolvere l'esercizio nello spazio bianco prima di aprire la soluzione.

---

⑦ Esercizio 1: Proprietà di Prefisso

Motivando la risposta, stabilire se i seguenti linguaggi godono della proprietà di prefisso. In caso negativo, fornire un controesempio.

1. $L_1 = \{a^n b^m \mid n > m \geq 1\}$
2. $L_2 = \{a^n b^n c^k \mid n, k \geq 1\}$

Tuo Svolgimento:

Primo Linguaggio

1. Provo il caso $m = 1, n = 2$
    1. $aab \in L, \quad aab \neq \varepsilon, \quad \varepsilon \notin L$
2. Aggiungo una b
    1. $aabb \notin L$
3. Provo $m = 1, n = 3$
    1. $aaab \in L$
    2. Aggiungo una b
    3. $aaabb \in L$
    4. $x = aaab \qquad y = aaabb$
    5. x è prefisso di $y \implies$ il linguaggio non gode della proprietà del prefisso

Secondo Linguaggio

1. Provo il caso $n, k = 1$
    1. $abc \neq \varepsilon, \quad abc \in L, \quad \varepsilon \notin L$

2. Aggiungo una c
    1. $abcc \in L$
    2. $x = abc \qquad y = abcc$
    3. x è prefisso di $y \implies$ il linguaggio non gode della proprietà del prefisso

✓ Soluzione >

1. L_1: NO.

- **Ragionamento:** Il numero di `a` deve superare le `b`. La coda di `b` è limitata da `n` ma variabile.
- **Controesempio:**
    - Scelgo $n = 3, m = 1 \Rightarrow x = aaab$. ($3 > 1$, OK).
    - Scelgo $n = 3, m = 2 \Rightarrow y = aaabb$. ($3 > 2$, OK).
    - $aaab$ è prefisso di $aaabb$.
- **Risposta:** Il linguaggio NON è prefix-free.

2. L_2: NO.

- **Ragionamento:** Abbiamo un blocco bilanciato $a^n b^n$ seguito da una coda libera c^k. Possiamo estendere la coda a piacimento.
- **Controesempio:**
    - Scelgo $n = 1, k = 1 \Rightarrow x = abc$.
    - Scelgo $n = 1, k = 2 \Rightarrow y = abcc$.
    - x è prefisso di y.
- **Risposta:** Il linguaggio NON è prefix-free.

? Esercizio 2: Costruzione DPDA

Dato il linguaggio $L = \{a^n b^m c^m d^n \mid n, m \geq 1\}$, costruire un **Automa a Pila Deterministico**.

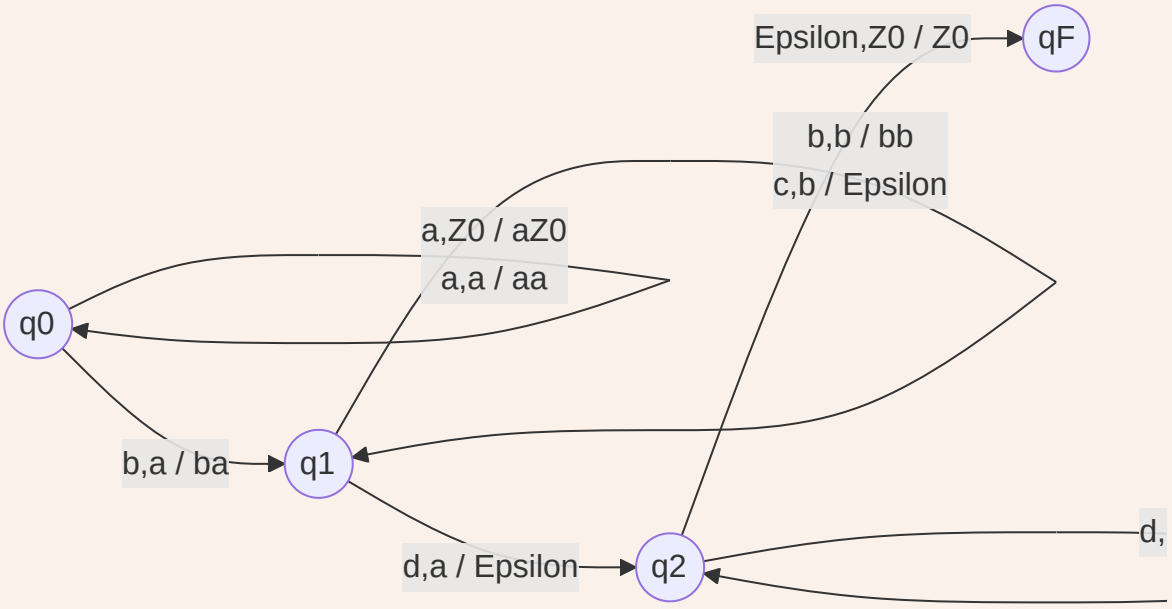
1. Definire la funzione di transizione δ.
2. Discutere la scelta della terminazione.
3. Scrivere la CFG che lo genera

Tuo Svolgimento:

Scelta e spiegazione del tipo di DPDA

. Controllo se L è prefix-free per decidere il tipo di DPDA
1. Caso $n, m = 1$
2. $abc \in L, \quad abc \neq \varepsilon, \quad \varepsilon \notin L$
5. Caso $n = 1, m = 3$
1. $abbbcccd$
2. Incremento n
3. $aabbbcccdd$
4. Non esiste x tale che $w = xy$
5. Il linguaggio gode della proprietà del prefisso
6. Posso usare DPDA per stato accettante o per pila vuota
7. **Scelgo stato accettante perchè è più facile**

Grafico del DPDA



✓ Soluzione ›

Analisi: Struttura a specchio annidato ("Nested").

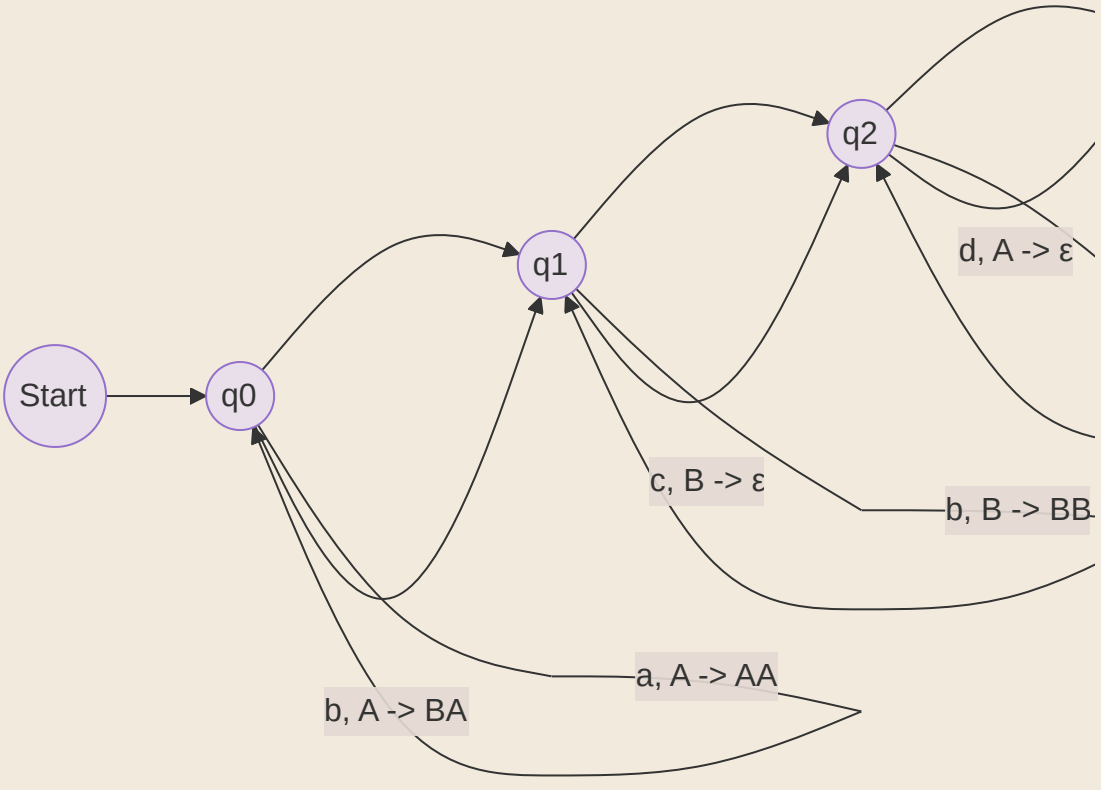
1. Leggo `a`, Push `A`.
2. Leggo `b`, Push `B` (sopra le A).
3. Leggo `c`, Pop `B` (Match 1:1 con le b).
4. Leggo `d`, Pop `A` (Match 1:1 con le a).

Funzione δ:

1. $\delta(q_0, a, Z_0) = \{(q_0, AZ_0)\}$
2. $\delta(q_0, a, A) = \{(q_0, AA)\}$
3. $\delta(q_0, b, A) = \{(q_1, BA)\}$
4. $\delta(q_1, b, B) = \{(q_1, BB)\}$
5. $\delta(q_1, c, B) = \{(q_2, \epsilon)\}$
6. $\delta(q_2, c, B) = \{(q_2, \epsilon)\}$
7. $\delta(q_2, d, A) = \{(q_3, \epsilon)\}$ (Cambio stato per gestire le d)
8. $\delta(q_3, d, A) = \{(q_3, \epsilon)\}$
9. $\delta(q_3, \epsilon, Z_0) = \{(q_f, Z_0)\}$

Terminazione: Il linguaggio è Prefix-Free (struttura rigida). Possibile usare Pila Vuota o Stato Finale indifferente.

Diagramma:



Esercizio 3: CFG to NPDA

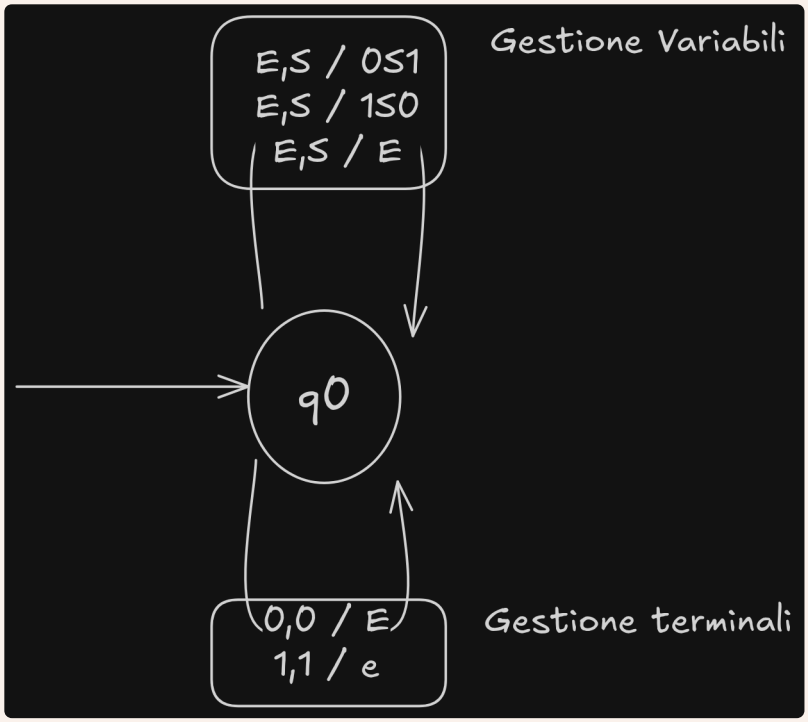
Data la grammatica:

$$S \to 0S1 \mid 1S0 \mid \epsilon$$

1. Costruire l'NPDA equivalente.
2. Mostrare la trace per la stringa `0110`.

Tuo Svolgimento:

PDA



Trace

$$(q_0, 0101, S) \vdash (q_0, 0101, 0S1)$$
$$(q_0, 0101, 0S1) \vdash (q_0, 101, S1)$$
$$(q_0, 101, S1) \vdash (q_0, 101, 1S01)$$
$$(q_0, 101, 1S01) \vdash (q_0, 01, S01)$$
$$(q_0, 01, S01) \vdash (q_0, 01, 01)$$
$$(q_0, 01, 01) \vdash (q_0, 1, 1)$$
$$(q_0, 1, 1) \vdash (q_0, \varepsilon, \varepsilon) \text{ Trace Completata}$$

✓ Soluzione ›

1. Costruzione (Stato unico q):

- $\delta(q, \epsilon, S) = \{(q, 0S1), (q, 1S0), (q, \epsilon)\}$
- $\delta(q, 0, 0) = \{(q, \epsilon)\}$
- $\delta(q, 1, 1) = \{(q, \epsilon)\}$

2. Trace `0110`:
Notazione: $(stato, input, stack)$

1. $(q, 0110, S)$
2. $\vdash (q, 0110, 0S1)$ (Espansione $S \to 0S1$)
3. $\vdash (q, 110, S1)$ (Match 0)
4. $\vdash (q, 110, 1S01)$ (Espansione $S \to 1S0$)
5. $\vdash (q, 10, S01)$ (Match 1)
6. $\vdash (q, 10, 01)$ (Espansione $S \to \epsilon$)
7. $\vdash (q, 10, 1)$ ... **Errore di trace**. La regola $S \to \epsilon$ rimuove S.
   Correzione Trace:
8. $\vdash (q, 10, 01)$ -> Qui lo stack ha `0` in cima, l'input ha `1`. **Mismatch**.
   Significa che ho scelto l'espansione sbagliata al passo 4.
   Riprova: Derivazione corretta $S \to 0S1 \to 0(1S0)1 \to 01(\epsilon)01 = 0101$.
   La stringa `0110` NON è generabile da questa grammatica (genera w t.c. $w = x^R$ negato sui bit? No, genera stringhe dove bit_i e bit_{n-i} sono diversi).

Trace corretta per stringa accettata `0101`:
9. $(q, 0101, S)$
10. $\vdash (q, 0101, 0S1)$
11. $\vdash (q, 101, S1)$
12. $\vdash (q, 101, 1S01)$
13. $\vdash (q, 01, S01)$

14. $\vdash (q, 01, 01)$ $(S \to \epsilon)$
15. $\vdash (q, 1, 1)$
16. $\vdash (q, \epsilon, \epsilon)$ -> Accettato.

Esercizio 4: Macchina di Turing

Data la MT con q_0 iniziale:

- $\delta(q_0, 1) = (q_0, 1, R)$
- $\delta(q_0, 0) = (q_1, 1, R)$
- $\delta(q_1, 1) = (q_1, 0, R)$
- $\delta(q_1, 0) = (q_0, 1, R)$

Mostrare la sequenza di configurazioni per input `110110`.

Tuo Svolgimento:

✓ Soluzione >

Trace:

1. $(q_0, \underline{1}10110)$
2. $\vdash (q_0, 1\underline{1}0110)$
3. $\vdash (q_0, 11\underline{0}110)$
4. $\vdash (q_1, 111\underline{1}10)$ (Trovato 0 $\to$ Scrive 1, cambia stato)
5. $\vdash (q_1, 1110\underline{1}0)$ (Trovato 1 $\to$ Scrive 0, resta in q_1)
6. $\vdash (q_1, 11100\underline{0})$ (Trovato 1 $\to$ Scrive 0...)
7. $\vdash (q_0, 111001\underline{B})$ (Trovato 0 $\to$ Scrive 1, torna in q_0)

Funzione: La macchina scorre a destra. Quando trova uno `0`, lo inverte in `1` e cambia "modalità". In modalità q_1, inverte `1` in `0`. Se trova un altro `0`, lo inverte in `1` e torna normale.

Esercizio 5: Teoria

1. Perché siamo certi che esista almeno un linguaggio NON Ricorsivamente Enumerabile?
2. Dare la definizione di L_d (Linguaggio Diagonale).

Tuo Svolgimento:

✓ Soluzione

1. Cardinalità:

- L'insieme delle Macchine di Turing è **numerabile** (possiamo enumerare le loro codifiche binarie $w_1, w_2, \ldots$).
- L'insieme di tutti i possibili linguaggi su un alfabeto Σ è l'insieme delle parti di Σ^*, che ha la cardinalità del continuo (**non numerabile**).
- Poiché ci sono "più" linguaggi che macchine, devono esistere linguaggi non riconosciuti da nessuna macchina.

2. Definizione L_d:

- $L_d = \{w_i \in \{0,1\}^* \mid w_i \notin L(M_i)\}$.
- È l'insieme delle stringhe w_i (che codificano la macchina M_i) che NON vengono accettate dalla macchina M_i stessa.

- L_d non è RE (dimostrazione per diagonalizzazione/assurdo).